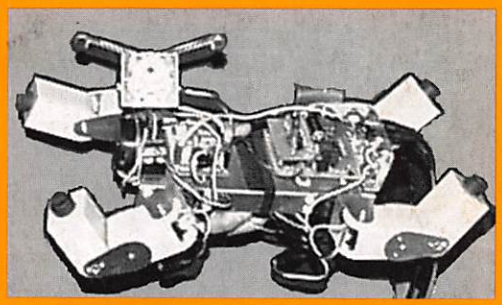
August 1999

;login:

The USENIX Association Magazine

volume 24 • number 4



inside:

REPORTS from the Workshop on Embedded Systems

SAGE NEWS & FEATURES:

Neil Kohl on Apache Modules
Mark Burgess on Cfengine
How-To: Configure DNS
Elizabeth Zwicky on SNMP

features:

The Tclsh Spot
by Clif Flynt

Source Code UNIX
by Bob Gray

Java Performance
by Glen McCLuskey

Using Java
by Prithvi Rao

Musings
by Rik Farrow

and more . . .

USENIX The Advanced Computing Systems Association

upcoming events

8th USENIX Security Symposium

WHEN	WHERE	WHO
August 23-26/99	Washington, D.C.	**Win Treese,** *Program Chair* **Avi Rubin,** *IT Coordinator*

2nd Conference on Domain-Specific Languages

Sponsored by USENIX in cooperation with ACM SIGPLAN and SIGSOFT

WHEN	WHERE	WHO *program chair*
October 3-6/99	Austin, TX	**Thomas Ball**

DEADLINES	
Final Papers	**Early Registration Discount**
August 24/99	September 13/99

2nd USENIX Symposium on Internet Technologies and Systems

Co-sponsored by the IEEE Computer Society Technical Committee on the Internet

WHEN	WHERE	WHO *program chair*
October 11-14/99	Boulder, CO	**Fred Douglis**

DEADLINES	
Final Papers	**Early Registration Discount**
August 31/99	September 17/99

3rd Annual Atlanta Linux Showcase

Co-sponsored by USENIX, Linux International and the Atlanta Linux Enthusiasts

WHEN	WHERE
October 12-16/99	Atlanta, GA

13th Systems Administration Conference (LISA '99)

Co-sponsored by USENIX and SAGE

WHEN	WHERE	WHO
November 7-12/99	Seattle, WA	**David Parter,** *Program Chair* **Phil Scarr,** *IT Coordinator* **Pat Wilson** *Practicum Track Chair*

DEADLINES
Final Papers
September 1/99

NordU2000/2nd Nordic EurOpen.se/USENIX Conference

WHEN	WHERE
February 8-11/2000	Malmö, Sweden

DEADLINES		
Paper Proposals	**Notification to Authors**	**Final Papers**
September 13/99	October 11/99	December 13/99

Tcl/2K: The 7th USENIX Tcl/Tk Conference

WHEN	WHERE	WHO *conference co-chairs*
February 14-18/2000	Austin, TX	**De Clarke & Tom Poindexter**

DEADLINES		
Paper Proposals	**Notification to Authors**	**Final Papers**
September 1/99	October 1/9	December 20/99

SANE 2000/2nd International System Administration and Networking Conference

Organized by NLUUG, co-sponsored by USENIX and Stichting NLnet

WHEN	WHERE	WHO *conference co-chairs*
May 22-25/2000	Maastricht, NL	**Bob Eskes & Edwin Kremer**

DEADLINES		
Extended Abstracts	**Notification to Speakers**	**Final Papers**
November1/99	November 12/99	March 24/2000

USENIX Annual Technical Conference

WHEN	WHERE	WHO
June 19-23/2000	San Diego, CA	**Chris Small,** *Program Chair* **John Heidemann & John Kohl** *IT Coordinators* **Kirk McKusick** *Freenix Track Coordinator*

DEADLINES		
Paper Submissions	**Notification to Authors**	**Final Papers**
December 6/99	January 25/2000	April 25/2000

9th USENIX Security Symposium

WHEN	WHERE	WHO *conference chair*
August 14-17/2000	Denver, CO	**TBD**

4th Symposium on Operating Systems Design and Implementation (OSDI 2000)

WHEN	WHERE	WHO *conference co-chairs*
October 23-25/2000	San Diego, CA	**Michael B. Jones & Frans Kaashoek**

DEADLINES		
Paper Submissions	**Notification to Authors**	**Final Papers**
April 25/2000	June 29/2000	August 31/2000

For a complete list of future USENIX events, access *http://www.usenix.org/events*

contents

2 IN THIS ISSUE . . .

3 LETTERS TO THE EDITOR

5 THE USENIX CROSSWORD PUZZLE

CONFERENCE REPORTS

6 Report on the Workshop on Embedded Systems

SAGE NEWS AND FEATURES

10 Wisdom of Words
by Tina Darmohray

11 Truth, Justice, and the Sysadmin Way
by Hal Miller

12 Sysadmin Tools Your Vendor Never Told You About – Revisited
by Mike Nemeth

13 The SAGE Report
by Gale Berkowitz

14 How-To: Configure DNS
by Chris deLongpre

19 Webmaster's Toolbox: Apache Modules
by Neil Kohl

26 Managing Network Security with Cfengine – Part 1
by Mark Burgess

29 On Reliability: Security and Reliability
by John Sellens

40 Enough SNMP to Be Dangerous – Part 2
by Elizabeth Zwicky

FEATURES

43 Optimal Peripheral Access Using Pipe-Based Double-Buffering
by Diomidis Spinellis

46 The Tclsh Spot
by Clif Flynt

50 Source Code UNIX: What's Your Data Worth?
by Bob Gray

55 Software Mini-Review: Red Hat Linux 6.0
by Bob Gray

56 The Trouble with Biometrics
by Christopher Calabrese

62 Java Performance
by Glen McCluskey

67 Using Java: Input and Output Streams
by Prithvi Rao

71 Musings
by Rik Farrow

STANDARDS REPORTS

75 An Update on Standards Relevant to USENIX Members

BOOK REVIEWS

79 The Bookworm
by Peter H. Salus

80 Handbook of Programming Languages
Reviewed by Glenn Vanderburg

USENIX NEWS

85 Member Dues and 1998 Financial Statements
by Gale Berkowitz

88 20 Years Ago in USENIX
by Peter H. Salus

88 USENIX Annual Awards

89 Board Meeting Summary
by Ellie Young

ANNOUNCEMENTS AND CALLS

90 2nd Conference on Domain-Specific Languages

92 2nd USENIX Symposium on Internet Technologies & Systems

94 2000 USENIX Annual Technical Conference

96 motd
by Rob Kolstad

Cover Photo: Boston, Massachusetts, with Lee & Seung's robot dog (see p. 8).

;login:

;login: is the official magazine of the USENIX Association.

;login: (ISSN 1044-6397) is published bimonthly by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$50 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $80 per year.

Periodicals postage paid at Berkeley, CA and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Editorial Staff

Editor:
Rob Kolstad <kolstad@usenix.org>

SAGE News and Features Editor:
Tina Darmohray <tmd@usenix.org>

Standards Report Editor:
Nick Stoughton <nick@usenix.org>

Managing Editor:
Jane-Ellen Long <jel@usenix.org>

Copy Editor:
Eileen Cohen

Proofreader:
Kay Keppler

Designer:
Vinje Design

Typesetter:
Festina Lente

Membership and Publications

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>
WWW: <http://www.usenix.org>

The closing dates for submission to the next two issues of *;login:* are October 5, 1999, and December 1, 1999.

in this issue . . .

by Jane-Ellen Long
Managing Editor
<jel@usenix.org>

There are issues and then there are ISSUES. Right off the bat, and across from this column, one of the latter is raised in a letter regarding the SAGE certification, uh, issue.

Further along, Peter Salus takes a slightly dyspeptic view of the recent Workshop on Embedded Systems: do you really want your tomatoes to talk back to you? And then there is the small matter of security: Mark Burgess starts a series of pieces on cfengine; John Sellens deals with anti-spam features in sendmail; and Chris Calabrese takes up the gauntlet thrown down by Dario Forte in the April *;login:* and suggests that biometrics spells trouble.

Want more? How about Linux? Both Rik Farrow and Bob Gray try to install Red Hat Linux 6.0, with some interesting results.

We love it! One of our dirty little secrets is the hope of seeing a few sparks fly when someone opens *;login:*. The tradition at USENIX events is to speak your mind, mercilessly question speakers, argue until the small hours in some BOF or other. Think of *;login:* that way: it's a forum where members can grind their axe, argue with their friends, and point out the errors of their ways.

All very politely, of course.

For those of you with more sedate tastes, allow me to point you in the direction of another of Peter Salus's Bookworm columns where he argues for shorter books followed by Glenn Vanderburg's review of Salus's own magnum opus – and I do mean *magnum.*

And for the truly serious policy wonks, we offer our annual pie charts demonstrating once again that we spend your dues money wisely.

Have a cool summer.

letters to the editor

SCSA Concerns

From Daniel Brockman

<brockman@pacificseries.com>

I have worked with computers professionally for 22 years, first as a programmer, and the last eight years as a systems administrator. I like systems administration. The work suits me well. I have a consulting business with satisfied clients and a promising future.

I have concerns about what the SCSA (SAGE Certified Systems Administrator) movement might do to that future. I write this in hopes others will consider these concerns. I have a fear that SCSA threatens aspects of my work that I presently enjoy.

SCSA would, I think, put a fence around the work. I wonder whether I will be on the inside or on the outside. Being outside and not easily able to get in could threaten my future income or force a career change. Being inside could limit me to a narrow range of tasks that I'd have to perform to the letter of a "professional" standard, and that could truncate my latitude, my success, and my pleasure in solving the ill-defined problems that come my way. SCSA could discourage technically creative solutions. SCSA could impose a legal liability on me if I deviate from the standard.

There are social aspects as well. Certification could stratify USENIX, so that only the SCSAs could write articles or letters for [the SAGE section of] *;login:*. As an SCSA, my clients might discount my perspective on business issues as impractically technical.

The uncertain rigors of the certification process concern me. In our general society, there are many certificates. A driver's license is a certificate almost any adult can get without too much work. This kind of certificate is not intended to exclude people, but to assert that they demonstrated enough knowledge about vehicles to operate one with some degree of safety. Other certifications, such as for CPAs or actuaries, have a strong economic barrier-to-entry aspect. Whatever the justice or the injustice in it, these certifications link up with legal requirements. Only the few who are certified have the privilege of carrying out particular tasks and the privilege of collecting the fee made larger by the difficulty of finding a certificate holder.

Proponents of certification say it assures the buyer of competence. But I did not choose my suppliers based on their certificates. My lawyer, my doctor, my dentist, my chiropractor, my carpenter, my realtor, and my mortgage lender all came through references. My own clients rely on my references. I don't think SCSA would change that.

I suppose there would be a test or tests for SCSA. I wonder what questions or demonstrations might appear on those tests. When I think on what I have done in my career, and the moments I would rather not relive, and how I accomplished what I did, and what I hope to do in the future, I think my success derives not so much from things I know but from my talent for figuring out things I don't know. I wonder whether a certification test would recognize that talent. I wonder how a test would show my ability to forge productive agreement among contending users. I wonder how a test would evaluate my responsiveness to changing conditions.

More likely, the test would survey my acquired arcana, and I would feel like Gulliver assaulted by Lilliputians who might even prevail. For example, my adventures have rarely led me to encounters with syslog. So I don't know much about syslog, and I've probably forgotten most of what I ever knew. However, on the two occasions when I had to make syslog work for me, I got it working as needed, and when I have to get syslog to do something in the future, I expect I will figure it out. For another example, there is emacs. I worked with emacs about 13 years ago for an extended period, and it worked for me on that non-UNIX system. I don't think I have used it since then. I have installed emacs a few times and I have gotten programmers started with it, but I use vi. I don't know much about emacs, and I am certain I will figure out whatever emacs I might want to know when I think it useful and important to do so. Will the SCSA test include questions on particular characteristics of emacs and syslog? If it does, then I can choose to study hard these topics which will add but marginally to my expertise, or I can choose to go uncertified, or I can study hard on syslog and emacs and get blindsided not knowing what service corresponds to port 17.

Naturally, some bureaucracy will arise around the certification process. There will be a committee to set the hoops through which we must all jump. We will fill out forms, and wait for cyclical processes, and pay fees, and correct the misspellings of our names, and hope it's all done for noble cause.

SCSA might cast me as a servant, not a role to which I aspire. Many people think of systems administration as a service. I prefer to think of it as an arbitration of contending desires, with me as the arbiter. I am nobody's servant. Will SCSA require me to swear an oath devoting myself to everlasting selfless service to the vice presidents of my client organization? Maybe that is a look in the wrong direction. The certification committee will consist of people who think SCSA is a good thing, otherwise they would not be on the committee. How many of them will see in it a fulfilling way to impose their own version of order? Though their intentions are probably most benign, will my certification serve to promote their fulfillment at my expense?

In summary, SCSA affords me no clear and obvious benefit. If I fail certification, it could force me to change careers. If I pass certification, it could encumber my creativity. SCSA could stratify USENIX, and fulfill the careers of the certification

more letters...

committee at my expense. It could limit my work to strictly technical issues and cast me in a servile role. SCSA may overlook my professional strengths and focus instead on relative minutiae. SCSA could create an economic barrier-to-entry and impose an administrative overhead, without affording practical marketable assurance of competence. These are my concerns.

I think SCSA may come inevitably. Many people want it. I suppose they think it will improve their lives and their future incomes. So, I suppose they will keep at it until SCSA is a reality. I am optimistic about my own future, and I don't see that SCSA could improve it. When SCSA comes, how will it change my life?

Three to the SAGE Editor

From Bruce W. Mohler

<bruce.w.mohler@saic.com>

I wanted to drop you a note to say thanks for letting me get my article on system summaries published in *;login:* (April 1999). To date, 51 people from all over the world have expressed interest in helping with the project and I'm hoping to package up and send out the source code today.

Again, thanks!!!

From Robert Yoder

<ryoder@tci.com>

I must take issue with your statement in the June 1999 *;login:* "A degree is not required to be a Professional Engineer."

Doing some research, I found, to my surprise, that this statement is technically correct, in California (see <ftp://leginfo.public.ca.gov/pub/code/bpc/06001-07000/6750-6766>).

But for most of the nation, it is false (see <http://www.ncees.org/>; choose "Engineers & Land Surveyors," bottom). Here's an excerpt:

> Licensed professionals have satisfied a number of important requirements on their way to professional licensure. While the requirements for licensure do vary somewhat from state to state, there is a great deal of similarity as well. In most cases, a successful candidate has completed a minimum of an undergraduate education and has successfully passed the Fundamentals examination. Most jurisdictions then require that the candidate has four years of experience working under the supervision of a licensed professional, followed by the candidate taking and passing the appropriate Principles and Practice examination. It's very important for you to know the requirements for licensure in your state or jurisdiction. Please contact your local licensing board for these requirements.

To look up info on individual state laws, see <http://www.ncees.org/engineers/licenseboard.html>.

Choosing Texas as an example (<http://www.main.org/peboard/law.htm>), here's another excerpt:

> Section 12. GENERAL REQUIREMENTS FOR LICENSURE. (a) The following shall be considered as minimum evidence satisfactory to the Board that the applicant is qualified for licensure as a professional engineer:
>
> (1) graduation from an approved curriculum in engineering that is approved by the Board as of satisfactory standing, passage of the examination requirements prescribed by the Board, and a specific record of an additional four (4) years or more of active practice in engineering work, of a character satisfactory to the Board, indicating that the applicant is competent to be placed in responsible charge of such work; or
>
> (2) graduation from an engineering or related science curriculum at a recognized institution of higher education, other than a curriculum approved by the Board under Subdivision (1) of this subsection, passage of the examination requirements prescribed by the Board, and a specific record of at least eight (8) years of active practice in engineering work of a character satisfactory to the Board and indicating that the applicant is competent to be placed in responsible charge of such work. . . .

From Bob Moneymaker

<Moneymaker@aiso.com>

What a refreshing point of view you expressed in your column [of June 1999]. To me your point made an awful lot of sense. It was interesting that you contacted the Board of Registration. I spent 13 years as a member of the surveying and engineering profession in California and the surveyors were going through the same wailing and gnashing of teeth that we are engaged in in the profession of system administration. How that debate came out or if it was even finished I don't know.

I do know that I hope more and more reasonable voices such as yours continue to surface so that we, as system admins, can continue to progress in being recognized as a separate group worthy of distinction in the IT industry.

Thanks again for your sensible comments.

Flavors of *BSD

From Tim Daneliuk

<tundra@tundraware.com>

In the latest issue of *;login:* Rik Farrow asked for comparison/contrast of the *BSD variants. Silly question – everyone knows God created the world using BSD and edited the relevant scripts with `emacs`. All other systems are infidel by nature and lesser by definition. The BSD variants are merely minor theological schisms along the path of The One True Way.

(rabid, er, I mean, *avid* FreeBSD user ;)

more...

Rik Farrow replies:

Tim:

Actually, I am interested in the minor differences between variants, and why they are there. Infighting between different UNIX factions is one of the reasons for Microsoft's success with inferior operating systems. Ideally, there are differences, and those differences lead to best-of-breed feature sets in *nix. But what I would rather not see are incompatibilities introduced mainly to make it difficult to port software, device drivers, and system administrators between the various *nices.

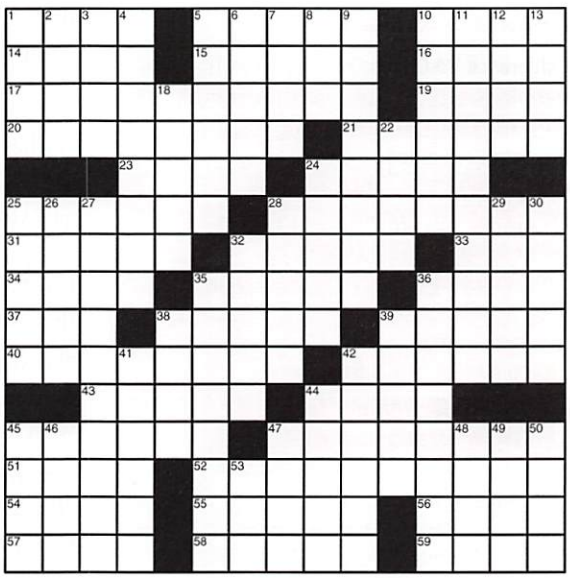
So, I am really looking for people who can explain to me some of the history behind there being four different versions of BSD today (net, open, free, and incorporated), how they are different, and how this helps us (or if this helps us).

Anyway, thanks for your letter.

S	P	A	T	■	C	A	S	A	■	U	S	U	R	P
K	A	V	A	■	A	R	A	B	■	N	E	P	A	L
I	L	E	X	■	N	A	I	L	■	S	M	O	T	E
D	I	R	I	G	I	B	L	E	■	T	I	N	E	A
■	■	■	■	A	N	Y	■	■	S	O	P	■	■	■
M	A	S	H	I	E	■	C	O	M	P	U	T	E	R
A	M	A	I	N	■	T	U	T	U	■	B	O	L	A
R	I	B	S	■	H	A	B	I	T	■	L	O	A	D
K	N	I	T	■	E	R	I	C	■	D	I	T	T	O
S	O	N	O	R	A	N	T	■	R	I	C	H	E	N
■	■	■	G	I	D	■	■	R	E	S	■	■	■	■
P	E	A	R	L	■	C	H	E	C	K	S	U	M	S
A	U	R	A	L	■	H	A	L	E	■	E	R	I	E
C	R	I	M	E	■	A	L	A	S	■	T	I	N	T
T	O	A	S	T	■	T	O	Y	S	■	A	C	T	S

Solution to the June Puzzle

The USENIX Crossword Puzzle

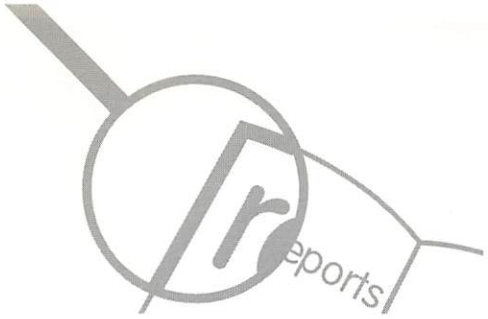


Across

1. Smell
5. Banana leaf fiber
10. Plant shell
14. Catholic leader
15. Word of mouth
16. Unknown auth.
17. Tree walking
19. Cask stave splitter
20. 3D round object
21. Marked by weakness
23. Hotels
24. N-way trees
25. Network conduits
28. Artists in oil
31. Dig these in to be stubborn
32. Anglo-Saxon advisor to king
33. 17th Greek Letter
34. Capri's land type
35. One of a Prussian cavalry body
36. Eagerly craving
37. Empty pointer
38. Speak scornfully
39. Secr.
40. Wrestled
42. Not often
43. Empty
44. Ole!
45. Purges
47. Pain and a tender feeling
51. NE St.
52. Having three teeth
54. Teen scourge
55. Devoured
56. Press
57. Lion's song
58. Colorers
59. Bonded with metal sheath

Down

1. Chooses
2. Village
3. Bright fish
4. Wake up call
5. Kitchen clothes
6. Foundation
7. Deodorant brand
8. Money-making scheme
9. Citizen of African country
10. Temple (Scot.)
11. Neither cautious nor reticent
12. Chimney filth
13. Patella's home
18. Sea eagles
22. Particular Scandinavian
24. Turk
25. Last imperial dynasty
26. Chief gods of Norse pantheon
27. Poisonous nightshade
28. Stacked
29. Horned animal
30. Destroyed Palestinian city
32. Few (obscure Brit.)
35. Not in the phone book
36. One ocean
38. Practice boxing
39. Star Trek engineer, for short
41. Trimmer
42. Bewitching Greek mythological creatures
44. Electronic speech maker
45. Wound's memory
46. Fibrous husk of a palm fruit
47. Locate
48. Viscount superior
49. Greek portico
50. Transmit
53. Gamma ____



conference reports

This issue's report focusses on the Workshop on Embedded Systems, held in Cambridge, Massachusetts, on March 29-31, 1999.

Our thanks to:

the summarizer:

Peter H. Salus

the Workshop Organizers:

Conference Co-Chairs:
Dan Geer, CertCo
Mike Hawley, MIT Media Lab

Program Committee
Mark D. Baushke, Cisco Systems
Warren Bosch, Hasbro
Tom Kalil, The White House
Tim Matt, Siebe
Jean Scholtz, DARPA
Ted Selker, IBM
Randy Sweeney, Kraft Foods
Jim Waldo, Sun Microsystems
Kevin Kelly, WIRED magazine

and to the MIT Media Lab!

Workshop on Embedded Systems

CAMBRIDGE, MASSACHUSETTS
March 29-31, 1999

Summary by Peter H. Salus

Though most folks aren't conscious of it, their homes and workplaces are full of computers – or at least parts of computers. Cars, bread machines, TVs, VCRs, coffeemakers, rice cookers, dish and clothes washers, etc., etc., are full of computing power.

The Workshop on Embedded Systems, co-sponsored by the MIT Media Lab and USENIX, spent three days considering the issues and the problems.

The wide success of Hasbro's "Furby" has demonstrated just how much we can do with a six-for-a-dollar chip.

Among the "fun" things considered were "things that think": intelligent refrigerators, pantries, and cutting boards, where food freshness and suchlike would be continually monitored. ("Your can of tomatoes should be used or thrown away within 10 days.") Just what happens to someone like me, who buys yogurt and keeps it weeks beyond its expiry, so that it separates and I pour off the whey to cook with the residue, is unclear.

One extremely interesting talk, though, was by David Smith, president and CEO of Object Automation, Inc. "The promise of object-oriented technology has long been that of reusable, portable software components," he said. "Contemporary computing technologies, like COM and CORBA, coupled with those that accompany the evolution of personal computing . . . the PC itself, plus networking and internetworking, have provided an environment where this promise has become a reality. The benefits that derive from the reality of truly reusable software components are enormous, even though much of that benefit is yet to be realized."

He emphasized factory improvements, as they spur the "fourth industrial revolution."

Talking to your box of breakfast cereal may not be in vain in the future. Whether it will be rewarding is another question. I had a vision of foodstuffs and other household products clamoring at me as I freewheel down the aisles of my supermarket. No, thanks.

Some of the graduate students at the Lab are experimenting with building circuits on paper with a Xerox machine. Metallic inks seem to work fine and (experimentally) they can do away with the chip in packaging. (They did a box of mix that responds to pressure on a list with the print of that specific recipe.) Perhaps there will be an electronically dense wrap so that we can cut off the packages' eyes and ears.

But the creation of neat things isn't all.

Session: Computer Everywhere

The Personal Node (PN)

G.G. Finn and Joe Touch, University of Southern California

PN is a small wallet-card that connects to the Internet. Among the points made were:

- Bodies on line: heart attacks not treated in time cost $130 billion a year.
- Remote needs: objects that can find us.

The most interesting questions and answers:

- How does one distinguish what's important? Simple set of buttons; half-size of PalmPilot; then shrink down.
- Geographic addressing? Perhaps GPS.
- Authentication for financial transactions? We'd like this – mapped to user.
- Policy issues: privacy; tracking. What does privacy mean? We don't know. What about anonymity?

- How does the device know good guys from bad guys? There are more than merely aggregation problems.

Several folks questioned the value of IR and RF links and one remarked that the authors were wrong on the IP issue.

Discourse with Disposable Computers: How and Why Will You Talk to Your Tomatoes

David Arnold, Bill Segall, Julian Boot, Simon Kaplan, and Melfyn Lloyd, DSTC

These guys have read too much Neal Stephenson. Picture Hiro's pizza box extrapolated to chewing-gum wrappers. Now, anticipate the clamor as you pass a trash can. Think of the event volume.

The most pressing discussion concerned the "noise" level; the notion of "content-based routing"; and the fact that a subscription model is dependent on receivers, not senders.

Smart Office Spaces

Bora Akyol, Alden Jackson, Rajesh Krishnan, David Mankins, Craig Partridge, Nicholas Shectman, and Gregory Troxel, BBN Technologies

This was an entertaining, "toy" paper. I was irritated by some sexist remarks ("a confident touch typist should be able to edit a file if she has access to a keyboard . . . ") in the paper, but they're not "content." It is over 30 years since I was introduced to the notion of the paperless office. I'm not holding my breath. However, the concept of "smart spaces" is fetching in a sci-fi movie, if not at large.

Scaling was the most important topic of discussion. Working within one office, or even a group of offices, lacks the real complexities of the world. Moreover, with a sea of devices, how much security can there be? Will we end up with "stealth management"?

Session: Networking Infrastructure

AirJava: Networking for Smart Spaces

Kevin Mills, NIST

This was a different presentation on smart spaces. Mills sees us as "wireless islands" in a "global wired ocean." He foresees "many kinds of multicasting."

He was queried about "active construction" of a combination between Linux and JavaVM. Mills said they were "beginning construction" in the summer.

Further questions involved:

- Widget replication – we need core functionality, not JavaVM that gets in the way when you start playing with Jini.
- The financial threshold: about $10.
- The NRC panel on power for the "dismounted soldier." We need new and better battery technology.

Finally, it was noted that none of the attendees was a "hardware guy," and that those are the folks who may be most important.

Bringing the Internet to All Electronic Devices

Chris Sontag and Michael Howard, emWare, Inc.

This paper began with an assumption that I thought was important and well-articulated: What's the ultimate value of networking all these devices? We must make certain that the cost involved is lower than the gain.

Considerations include processor performance, power, memory, storage, and network limitations. Understanding costs and the problems involving end-user education are very important.

Questions that struck me:

- How do we change program locations? We reload portions or whole applications.
- How do we program? What sort of APIs can we have? We definitely need very flexible products.
- Security? With so many devices isn't physical security a problem, too? There's much work to be done.

RETHER: A Software-Only Real-Time Ethernet for PLC Networks

Tzi-cker Chiueh, State University of New York

RETHER is a software-only realtime Ethernet for networks made up of programmable logic controllers. I really liked this paper because it constrained its domain: PLCs are used overwhelmingly in industrial automation. At the same time, the software involved must be both fault-tolerant and not socket-based.

The discussion was quite animated. The use of single shared media assumes a point-to-point link, and whether each node needs to know which other node is using RT was among the topics. The speed of bandwidth growth was another subject. This led to the question of the capacity/performance curve and whether service guarantees were feasible. The final queries dealt with statistical loading and efficiency.

Session: Design

Pebble, A Component-based Operating System for Embedded Applications

Eran Gabber and Christopher Small, Lucent Technologies/Bell Labs; John Bruno, University of California, Santa Barbara; José Brustoloni and Avi Silberschatz, Lucent Technologies/Bell Labs

Pebble is an OS designed for high-end embedded communications devices. It comprises a set of replaceable, user-level components. There are benchmark results that appear to demonstrate Pebble's effectiveness.

- What about priorities? Done inside the scheduler – interrupts are converted to

schedules and delivered immediately, but not acted upon immediately.

- Portal generation is a one-time event.
- Is portal generation interruptable? Only when you are placing into the table.
- Untrusted components: you can still spoof the system. Software checking is needed.
- The scheduler knows about all the threads: "If you can't trust your scheduler, who can you trust?"

Massively Distributed Systems: Design Issues and Challenges

Dan Nessett, 3Com Corporation

Nessett envisions systems involving billions of nodes; the paper discusses the engineering problems this entails. The discussion included questions about unifying principles, measurement, testing and debugging, etc.

- What about the health industry? It's never an early adopter.
- Is there a way to control complexity through software? Central control is an impossibility.

Session: Control

Learning in Intelligent Embedded Systems

Daniel D. Lee, Lucent Technologies/ Bell Labs; H. Sebastian Seung, MIT

Lee and Seung have built a robot dog that learns to identify sounds and faces. It looks great, does its thing, and is built out of $700 worth of off-the-shelf parts. The mathematics was worth the price of admission. Watching the "dog" "walk" and search and identify was very impressive. It's the perfect pet for a small apartment.

Using Mobile Code Interfaces to Control Ubiquitous Embedded Systems

Kari Kangas and Juha Röning

All the code appeared to be in Java; insecurity is still a major drawback. The discussion focused on:

- different kinds of mobile-code protocols
- comparisons with Inferno involving the fact that this avoids using only one language (37KB of code)
- uses of code standardization

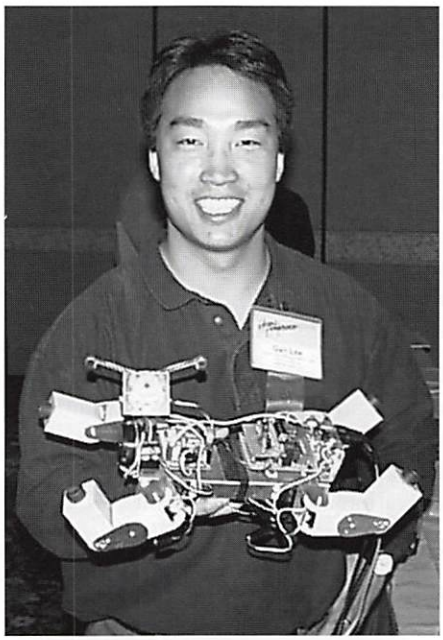
Challenges in Embedded Database System Administration

Margot Seltzer, Harvard University; Michael Olsen, Sleepycat Software

This was a really interesting paper because of the topics covered and the points made.

While the paper per se involved database architecture and the problems entailed by administration of embedded databases, perhaps the best portion of the paper was its list of aphorisms, some of which were:

1. SQL sucks.



Daniel D. Lee & Friend

2. Size matters.
3. Speed matters.
4. You don't need the whole Swiss army knife.
5. Service, service, service.
6. You get paid a lot, your life should be difficult; but the software should be bulletproof when it's rolled out.
7. Underestimate your user.
8. People hate surprises.
9. Resilience in the face of failures is important.

I'll stop here. The discussion involved the sheer size of some of the code; that complexity is still something we don't have a handle on; and that by-and-large humans adapt to failure.

Discussion

What Have We Learned? Where Do We Go?

Dan Geer, CertCo

Workshop co-chair Dan Geer led a (to me) interesting and important discussion concerning a variety of points central to the notion of the future of workshops like this one.

One participant, for example, said that he felt the workshop had been too diffuse, "all over the place," and (though interesting) had contained nothing he found of value. There were so many pieces – networking, security, ubiquity, control, robustness, etc., etc. – that he was like the blind men and the elephant.

Geer asked, "Who or what is missing?" Hardware guys, chip designers, etc., appeared to be the most obvious. The DNS folks and the IPv6 folks were also absent.

It was obvious that while no one was willing to define what an embedded system is, everyone thought they'd know one when they saw one. Even the question of what exactly a device is was brought up.

After a discussion of resilience and recovery, "Do we standardize?" was asked. Co-chair Mike Hawley indicated a strong "no." "Standards are an Ice Age," he said.

It became clear that we should become accustomed to living with incoherence (identity, language, scope of detection); that important questions were: "how do I talk to you?" and "explain yourself."

There was a lot of talk about hybrid vigor. Diversity is important in development. This vigor needs to be felt in domestic technology, factories, transportation, and office services. If it works (well enough), it will get installed.

The workshop concluded with a proposal that it be done "again, only better." Make it three days once a year or two days twice a year, preferably the latter. Keep the joint Media Lab / USENIX / Cambridge venue until we have a reason not to. Each morning is a case study of an actual deployed system – invited on the basis of its known relevance and using demos if at all practical. Make the afternoon some mix of theoretical results and general argument, possibly broken out into topic groups. Keep the number of participants small but get reps from the areas that should have been there this time but were not. Keep the workshop style as long as possible. Have, if someone will do it, some organized sense of what is unsolved (such as whether every device should be required to "explain itself" on demand) and keep a scorecard on progress. Pay as much attention to what has failed in the market (of ideas or of money) and why. Keep a place for refereed work (so those with a need for formal bibliography growth can participate) but have no qualms about inviting the rest of the event. Continue to have meals together but arrange both meeting and eating spaces to maximize conversation. Evolve into side-by-side events if and only if specialization is essential to progress. Geer: "Think big – somebody will and why not us?"

SAGEnews & features

Wisdom of Words

by Tina Darmohray

Tina Darmohray, editor of SAGE News & Features, is a consultant in the area of Internet firewalls and network connections and frequently gives tutorials on those subjects. She was a founding member of SAGE.

<tmd@usenix.org>

Hindsight is 20/20. The best defense is a good offense. Forewarned is forearmed. There's no time like the present.

Do you ever find yourself dispensing these well-known phrases as advice? I do. Sometimes I give them to others; sometimes I'm the recipient, as I lecture myself. I find that the advice these words impart is as important as the sheer fact that they exist. For example, "hindsight is 20/20" wouldn't mean nearly as much if it weren't applicable to so many folks at so many times. It's the one that helps me remember that I'm seldom the *only* one in this predicament, and that's consolation in itself at times. Bottom line, there's a lot of life wisdom delivered in these deceptively simple sentences. And, I'd argue, there's a lot of sysadmin advice in them as well.

The best defense is a good offense.

You will never be better prepared to thwart an attack, function in a crisis, or recover from a catastrophe than if you prepare for them ahead of time. I know that we've all heard this before, but sadly there are many sites, even high-profile organizations, that aren't heeding this advice and deploying industry best practices in system administration. It sounds simple, but there is a good reason that they are "best practices." You'll never regret spending system-administration cycles on basics like:

- creating and verifying backups ["It's not a backup until it's been restored."];
- limiting services and access;
- applying patches;
- using nonreplayable passwords over untrusted networks;
- considering encryption for remote access;
- having a policy;
- having a crisis plan and testing it to see if it is practical.

Hindsight is 20/20.

Of course, experience makes us all better. There's no need to apologize for learning from mistakes. If you didn't prepare in advance, let the lesson be that you never get caught off guard again. In all the disasters, break-ins, and computer incidents I've seen, rarely is there something truly new and different. Almost always it's a known problem that's been exploited and poor system administration that's at fault – and best practices would have prevented it.

Forewarned is forearmed.

I said rarely. Recently I've come across a so-called "dead-man switch." It is installed on compromised hosts to cause damage if it detects the host owner taking action to reclaim the machine. Because of this, it might warrant considering if you need backups before you "pull the plug" on a suspect machine, as such action might trigger a dead-man switch.

There's no time like the present. So, go out and get your site in order. It'll keep the bad guys out, the machines up, and you asleep at night! Remember, you reap what you sow.

Truth, Justice, and the Sysadmin Way

by Hal Miller

Hal Miller is president of the SAGE STG Executive Committee.

<halm@usenix.org>

The Australian Senate recently passed a bill that legislates liability onto ISPs for the content of data in use by their customers. SAGE-AU, among other computing (and other) societies in Australia, has taken a stand against this. Not all that long ago, they needed to do the same thing on another bill. Their official submission was quoted extensively in Parliament and their position accepted.

A couple of years ago here in the U.S., we faced a similar problem with the Communications Decency Act. SAGE is organized differently from our Australian cousins. Our laws (tax laws in particular) are different, and our ability to mobilize and act was "confused" at best. We were fortunate in that a court stopped enforcement of that act, but we actually could still face the problem – the law remains on the books. Another court could still decide against our interest, with great potential harm to our members.

It is probably safe to say that most legislators know little about computing technology. It is probably safe to say that most legislators (at state as well as national levels) would love to be able to report to their voters that they have "solved" the various pornography, gun control, or other social problems by passing a law forbidding the effects of these issues on the Internet. It seems safe to say that we will see a continuing spate of misguided although probably well-intentioned efforts to do this. The implication is that unless someone guides these efforts, sysadmins are going to find themselves in court, if not in jail, as scapegoats for various social problems.

The question I bring to you now is: What ought SAGE to be doing in this regard?

Obviously, there are a lot of sub-questions here, such as: What are we allowed to do? What organizational changes might we need to consider to widen our scope if we wished to do so? Where do the resources come from? Why would anyone want to listen to us? These questions are certainly important, but unless we have some idea of what we want to do, knowing what we "can" do, or how we go about it, doesn't seem important. While some may argue "cart before horse," especially in the "what we can do" area, I claim that the fact that we cannot bring Rembrandt back to life isn't relevant if we as an organization aren't worried about Dutch artists. First issue is: What do we want to do here?

Many things quickly spring to mind: congressional liaison, either as registered lobbyists or as recognized experts that legislators come to know and call upon for their subcommittee hearings; newspaper editorials and/or columns; radio talk shows (in the form of Internet-based communications, e.g., USENET news, email lists, Mbone); public education (PBS shows?, community-college courses), including for legislators and judges. But first again, what are we after? These are "implementation" ideas – what is the goal?

I believe that there is nobody out there who can (or will) act to protect us, the sysadmins, other than SAGE. I believe that it is unlikely anyone else can or will step forward with the knowledge level required to protect the innocent (e.g., ISPs for content-monitoring of customer users, recipients of spam pornography, etc.) from well-intentioned (or not) laws that have significant, negative, unintended results. (Where is Jack Webb when you need him?) SAGE needs to brush down its white charger, shine its armor and sword, and start searching for dragons to slay in this field. We do, of course, need

to ensure we don't start jousting with windmills or becoming the Sir Rodney (character in the newspaper comic "The Wizard of Id" who loses every jousting match) of the computing world, but there are issues out there that cannot be left to inertia, and which nobody else is suited to address.

The goal should be, therefore, to commit ourselves to playing Zorro from time to time, rather than ignoring these ever-increasing problems of the maturation of the Internet. I would like to hear your reactions to this statement. Please email me with thoughts and comments. (<sage-members> is available should you prefer public discussion.) "How-To"s are also solicited, both for the new series and for this!

If we accept this goal as a direction for SAGE, it becomes appropriate to begin with the other questions. I will touch briefly here on a few of them and would like to charter a group to tackle this in earnest.

What can we do? Clearly legislative lobbying makes sense. Problems abound with this, though, and if we choose it as a course of action, we will need to do a fair bit of preparatory work. SAGE is a piece, without independent legal standing, of a not-for-profit corporation (USENIX). Tax (and other) laws limit the kind and amount of lobbying USENIX can do as things stand now. There are ways to do some "lobbying," as we found during the CDA discussions, but we would need additional legal work and organizational "definition" (meaning addressing some hard internal questions we have successfully avoided thus far) in order to take more significant action. Maybe we can manage to get a few of our number elected to the various legislatures? (No, I'm not running!)

Another direction we could potentially work in is expanding our educational goals to a wider horizon. All along I have been thinking of educating sysadmins, but maybe we also need to find ways to educate the general public (or at least judges and legislative staffers) on the impact and fallout of the things they're doing in trying to achieve social goals.

In any event, for us to be effective in this arena (as with others) we need to significantly expand our market penetration. We need to add a couple of zeros to our membership numbers. This we can tackle without running afoul of tax laws, so I declare open season for recruiting. Join a local group if you're not in one, form one if there isn't one in your area, and start drawing individuals, companies/institutions, and vendors into your meetings!

Sysadmin Tools Your Vendor Never Told You About – Revisited

by Mike Nemeth

Mike Nemeth is an HPUX system administrator currently working at the Naval Air Warfare Center in Lakehurst, New Jersey. He has been a UNIX system admin for over eight years.

<mjnemeth@hotmail.com>

Elizabeth Zwicky's articles starting in the *;login:* July/August 1993 issue have provided me with some invaluable "tools" for keeping my sanity as a system administrator. My "Squeaky Dragon" has amused (and puzzled) users seeking service. Some gleefully squeak Buford (that's his name) and others are not sure what to do when they see the sign reading "Squeeze Dragon for Service!" My Jibberjaber, Biff (he is a dog), has provided me with needed stress relief. Sometimes you need to throttle something! Co-workers (fellow SAs and users) ask to borrow him; a few even get the joke about his name. I once bought two dozen when Jibberjabers were available. (Does anyone know where I can get more?) All but a few were immediately placed in good homes in the large SA group that I was working with then!

Now it is my turn to provide some invaluable SA "tools" that I have found useful over the years.

Users always seem to have a strong opinion on how the system should be managed or used or what the SAs should do; the list seems never-ending! Well, I have a large plastic jar with a slot in the lid. It is labeled "Put Your 2 Cents In!" When a user starts to "vent" opinions, I point at the jar. After a while I do not even need to point anymore but can hear "clink clink" and turn to find a smiling user. Now, every once in a blue moon, a user *does* have a good idea. When this occurs, I turn the jar around – this side is labelled "A Penny for Your Thoughts!" – and tell the user to take one!

Stress relief is essential! Here are instructions for putting together your own stress-relief kit. Take a standard 8.5 x 11 sheet of plain white paper and at the very top, in large letters, write the title: "ANTI-STRESS KIT." Just below this draw a large circle; it should almost touch the top title and the sides. Just below the circle, in large letters, insert the word "DIRECTIONS." Add the following directions:

1. Place on FIRM surface.
2. Follow directions provided in the circle on front of kit.
3. Repeat until you are anti-stressed or become unconscious.

Now in *large* letters in the center of the circle put the following:

BANG

HEAD

HERE

The next tool is not as funny as it first sounds, a rear-view mirror for your workstation or terminal! How many

times have you nearly jumped out of your seat (stress, remember?) when someone quietly walked up behind you and spoke?!

There are many other "tools" that I'll save for another time – like hats. (How many do you wear? SA, CM, CNN . . .)

The SAGE Report

by Gale Berkowitz

USENIX Deputy Executive Director

<gale@usenix.org>

Three meetings of the SAGE Executive Committee have been held since January 1, 1999. The most pressing issues for SAGE have been the SAGE Certification Project and the longterm strategic direction of SAGE.

SAGE Certification

The SAGE Certification Project is progressing well. Phase 1 was used to solidify project goals, plans, and procedures associated with the project as a whole. This phase has been completed.

We are now in Phase 2, the Occupational Analysis. This phase is viewed as supporting certification development, as well as education and training. It includes making use of available materials, incumbent interviews, and workshops. A survey is being conducted of job incumbents in order to build a description of the core requirements for system administration. The information derived from the occupational analysis will be used to determine the scope and content of Phases 3 and 4.

Two successful focus groups were conducted at the SANS conference in May in Baltimore. Two more are being organized in the California Bay Area, and another in the D.C. Metro area. Information derived from the focus groups will be used to develop the Web-based occupation survey to be conducted in July. Next steps include processing the information from the focus groups, interviewing members of the SAGE Certification Advisory Committee, and then developing and implementing a Web-based survey of systems administrators. We hope to launch the survey in September of this year.

Planning

Membership in SAGE continues to grow. The SAGE Executive Committee is holding a series of strategic planning meetings to map out the future direction of SAGE.

Other Ongoing SAGE Projects include:

How-To Notes

The How-To Notes are intended to be an ongoing, living, Web-based series. A How-To Notes Editor, Melissa Binde, has agreed to serve as the volunteer editor of the series. The first two Notes have been published in *;login:*. The third appears in this issue, and other notes are being reviewed. SAGE encourages new ideas for How-To Notes, as well as volunteers to write the notes.

Mentoring Project

The purpose of the SAGE Mentoring Project is to guide and encourage apprentices in the development of all skills necessary for success as a UNIX or NT system administrator. It aims to develop leadership and mentorship skills among the mentors as well. It is a one-year pilot project. The Mentoring Pilot Project, under the direction of Mike Ewan, is moving forward, and apprentices have been selected.

Call for How-To Authors and Reviewers

by Melissa Binde

How-To Series Editor

<binde@amazon.com>

Needed: knowledgeable systems administrators to write "How-To" documents forless experienced admins.

Goal: To provide an indexed, searchable, list of common sysadmin tasks with enough information to accomplish the task and enough pointers to other sources of information to do more research.

Target audience: Administrators who need to do something quickly and solidly and don't have the experience or time to learn it completely first.

Justification: The job market grows and the candidate pool shrinks. More and more admins are put into positions where they are asked to do things they haven't done before, without enough time to properly learn it. USENIX can provide a way for these admins to get enough up to satisfy their bosses,while still being technically correct, and pointers on where to go to learn more about the subject.

how-to

Configure DNS

by Chris deLongpre

Following many years as a network and UNIX administrator, Chris is now an Information Security Analyst at Metropolitan Health Corp., Grand Rapids, MI.

<Chris.deLongpre@metrogr.org>

This is not meant to be the definitive work on DNS, BIND, or named. For that we have *DNS and BIND* by Paul Albitz and Cricket Liu (O'Reilly & Associates, 1998). That fine book was actually the starting point for my first DNS configuration, for AIX v3.2.5. Following that successful experience, I developed some very simple documentation to explain the configuration to other system administrators at my site. When we migrated our main applications to an HP-UX environment, I developed simple documentation to explain DNS configuration under HP-UX.

The first two sections below cover a fairly straightforward and simple DNS configuration. You will note both similarities and differences between the AIX implementation and the HP-UX implementation. Both implementations are also in some ways similar to "standard" BIND (as described in the Albitz and Liu book) and in some ways different. A third section of this How-To takes the configuration a bit further by adding "forwarders" to forward requests to authoritative servers in another domain.

Although I originally wrote this How-To for AIX v3.x, I have since heard from admins who have successfully used this documentation to jumpstart DNS on AIX v4.x hosts. This document is meant to be only a starting point. It does not cover every situation. If, however, you need to get named configured and working in a hurry, this will help. You can embellish from that point to your heart's content.

1.00 Configuring DNS for AIX (v3.2.5)

1.10 Primary Nameserver (AIX):

The configuration of the primary nameserver consists of an empty `/etc/resolv.conf` file (the indication that hosts are resolved via DNS rather than `/etc/hosts`), an `/etc/named.boot` file (essential for running TCP/IP's named), and a set of database files in `/etc/nameserver` (`db.data`, `db.rev`, and `db.local`).

1.11 Make the nameserver directory:

```
mkdir /etc/nameserver
```

1.12 Update the `/etc/hosts` file so that all hostnames, IP addresses, and aliases are current and correct. This file is used as the basis for compilation of the DNS database files.

1.13 Create the `/etc/named.boot` file (with an editor). Sample contents of `/etc/named.boot`:

```
directory   /etc/nameserver
domain      [your domain name]
primary     [your domain name]          db.data
primary     in-addr.arpa                db.rev
primary     0.0.127.in-addr.arpa        db.local
```

1.14 Run the AIX scripts that will create the database files from `/etc/hosts`:

```
cd /usr/lpp/tcpip/samples
./hosts.awk /etc/hosts          /etc/nameserver/db.data
./addrs.awk /etc/hosts          /etc/nameserver/db.rev
```

1.15 Create the `/etc/nameserver/db.local` file (with an editor). Sample contents of `/etc/nameserver/db.local`:

```
@ IN  NS [your hostname].[your domain name].
1 IN  PTR localhost.[your domain name].
```

(Note: Trailing dots on fully qualified hostnames are mandatory.)

1.16 Create an empty `resolv.conf` file:

```
touch /etc/resolv.conf
```

1.17 Change hostname to reflect fully qualified domain name:

```
smit mkhostname
```

1.18 Start named services:

```
startsrc -s named
```

If this is to be a permanent primary nameserver and named should be started each time the system is rebooted, uncomment the named start line in /etc/rc.tcpip.

1.19 Test name resolution: Use the host command or nslookup.

1.110 Save a copy of the /etc/hosts file that was used in compiling the DNS database files (/etc/hosts.save). DNS does need an /etc/hosts file, but it only needs to contain the loopback record and should also contain a record for the local host. Sample contents of /etc/hosts:

```
#/etc/hosts
127.0.0.1           loopback            localhost
[local IP address]  [hostname]          [alias]
```

Retest name resolution after creating this abbreviated /etc/hosts file.

1.20 DNS Clients (AIX):

1.21 Create an /etc/resolv.conf file (with an editor). Sample file contents are:

```
domain      [your domain name]
nameserver  [IP address of primary nameserver]
```

1.22 Change the client's name to the fully qualified name:

```
smit mkhostname
```

1.23 Test name resolution: Use the host command or nslookup.

1.24 As a final test, revise /etc/hosts. This file needs to contain only the loopback record and a record for the local host, as illustrated under 1.110, above. Retest name resolution.

1.30 Secondary Nameserver (AIX):

A secondary nameserver needs to run named and so also needs an /etc/named.boot file.

1.31 Sample contents of the /etc/named.boot file on a secondary server:

```
directory /etc/nameserver
domain    [your domain name]
secondary [your domain name][IP add. of primary]   db.data
secondary [your domain name][IP add. of primary]   db.rev
primary   0.0.127.in-addr.arpa                     db.local
```

1.32 Create the db.local file. Sample contents:

```
@ IN NS [your hostname].[your domain].root.[your hostname].[your
                        domain].
(    1:1         ; serial
     3600        ; refresh
     600         ; retry
     3600000     ; expire
     86400 )     ; minimum
IN   NS [your hostname].[your domain].
1 IN PTR localhost.
```

(Note: Trailing dots on fully qualified hostnames are mandatory; there are four in this sample.)

1.33 Create the /etc/resolv.conf file:

```
touch /etc/resolv.conf
```

A secondary nameserver needs to run named and so also needs an /etc/named.boot file.

If this is to be a permanent secondary nameserver and named should be started each time the system is rebooted, uncomment the named start line in `/etc/rc.tcpip`.

1.34 Change the secondary nameserver's hostname to the fully qualified name:

```
smit mkhostname
```

1.35 Start named services:

```
startsrc -s named
```

If this is to be a permanent secondary nameserver and named should be started each time the system is rebooted, uncomment the named start line in `/etc/rc.tcpip`.

1.36 Test name resolution: Use the `host` command or `nslookup`.

1.37 Revise `/etc/hosts` as shown in 1.110, above, and retest name resolution.

1.38 Modify the `/etc/resolv.conf` files of all clients to include this new nameserver:

```
domain          [your domain name]
nameserver      [IP address of primary]
nameserver      [IP address of secondary]
```

2.00 Configuring DNS for HP-UX (v10.x)

2.10 Primary Nameserver (HP-UX):

2.11 Create the nameserver directory:

```
mkdir /etc/nameserver
```

2.12 Change directory to `/etc/nameserver` and edit a new file called `param`. See a man of `hosts_to_named` for an explanation of the options to place in this file. Sample contents:

```
-d  [your domain name]
-b  /etc/named.boot
-n  10.1.3 (your subnets)
-n  10.1.4
-n  10.1.7
-n  10.1.10
-n  10.1.22
-z  10.1.3.19 (where the secondary server will get its information)
-Z  10.1.3.19 (needed for a secondary nameserver)
```

2.13 Make sure `/etc/hosts` is current and all information is correct.

2.14 Create the database files. In `/etc/nameserver`, run:

```
/usr/sbin/hosts_to_named -f param
```

(Note: `/etc/named.boot` and all the database files in `/etc/nameserver` get created when this command is run.)

2.15 Edit `/etc/rc.config.d/namesvrs` and change `NAMED=0` to `NAMED=1`.

2.16 The DNS nameserver must be a client to itself. Create an `/etc/resolv.conf` file:

```
domain      [your domain name]
nameserver  [IP address of the primary nameserver]
```

2.17 Start named:

```
/sbin/init.d/named start
```

2.18 Test name resolution: Use `nslookup`.

2.19 Revise the contents of `/etc/hosts` (saving the master to a different name) as in section 1.110 above; retest name resolution.

2.20 DNS Clients (HP-UX):

2.21 Create an `/etc/resolv.conf` file on each client:

```
domain        [your domain name]
nameserver    [IP address of the primary nameserver]
```

2.22 Test name resolution: Use `nslookup`.

2.23 Revise the contents of `/etc/hosts` as in section 1.110 above and retest name resolution.

2.30 Secondary Nameserver (HP-UX):

2.31 Make the nameserver directory on the secondary server:

```
mkdir /etc/nameserver
```

2.32 Use ftp to the primary nameserver to get `/etc/nameserver/boot.sec.save` to install locally as `/etc/named.boot`. (Note: If you want the secondary server to store the data files on its disk, then you have to use the file `boot.sec.save` for `named.boot`. If the data should not be stored locally, you have to use `boot.sec`. I believe the redundancy of storing a copy of the database files on the secondary server is important, so I use boot.sec.save on the primary as `named.boot` on the secondary.)

2.33 Modify the `/etc/resolv.conf` file on this, the primary, and all other clients to include the secondary nameserver:

```
domain        [your domain name]
nameserver    [IP address of primary]
nameserver    IP address of secondary]
```

2.34 Edit `/etc/rc.config.d/namesvrs` and change `NAMED=0` to `NAMED=1`.

2.35 Start named:

```
/sbin/initd./named start
```

2.36 Test the secondary server:

a. Stop the named on the primary nameserver:

```
/sbin/init.d/named stop
```

b. On the secondary nameserver, and/or a client, issue an `nslookup`. The first stanza of the returned data should indicate that resolution took place through the secondary nameserver.

2.40 Maintaining DNS (HP-UX):

Maintaining DNS requires the record `/etc/hosts` file (normally kept on the primary nameserver as either `/etc/hosts` or `/etc/hosts.save`) be updated with any changes (adds, deletes, modifications). Then the database files are rebuilt. They are automagically pushed to the secondary nameserver.

2.41 Make modifications to `/etc/hosts` or `/etc/hosts.save` file on the primary nameserver. Copy `/etc/hosts.save` to `/etc/hosts`.

2.42 Rebuild the database files:

```
/usr/sbin/hosts_to_named -f param
```

2.43 Test resolution to a changed host / IP address.

2.44 Check to make sure the change was propagated to the secondary nameserver:

I believe the redundancy of storing a copy of the database files on the secondary server is important, so I use `boot.sec.save` on the primary as `named.boot` on the secondary.

The series number in the db.[domain] file on the secondary should match that on the primary. Below are samples taken from my primary and secondary files. The series number is 48 (I had updated the hosts table 47 times since I first implemented DNS on HP-UX at this site!).

`$head db.xx` (on primary nameserver):

```
@ IN        SOA [host].xx. root.[host].xx. (
                                    48      ; Serial
                                    600     ; Refresh
                                    3600    ; Retry every hour
                                    604800  ; Expire after a week
                                    86400 ) ; Minimum ttl of 1 day
  IN        NS  host.xx.
localhost   IN  A                   127.0.0.1
```

`$head -3 db.xx` **(on secondary nameserver):**

```
$ORIGIN .
xx IN       SOA [host].xx. root.[host].xx. (
                                    48 600 3600 604800 86400 )
```

2.45 You can force an update of the secondary server's information by restarting named:

```
/sbin/init.d/named stop
/sbin/init.d/named start
```

Material for the this section comes, in part, from Michael Robinton <petals@girlswear.com>. His sample files in "How to set up 'named'" were instrumental in determining the success of my forwarders configuration.

3.00 Adding Forwarders to the HP-UX (V10.X) Configuration

3.01 Stop `named`:

```
/sbin/init.d/named stop
```

3.02 Initialize the db.cache file (`/etc/nameserver/db.cache`). This file is created by `hosts_to_named` without any meaningful guts and is overhead until you need to use it. Sample of revised `db.cache`:

```
;
.           99999999        IN NS host.somewhere.com
host.somewhere.com.         99999999      IN A 10.1.3.25
;
```

Here you put the fully qualified hostname and the IP address of the root nameserver to which you wish to forward requests for resolution outside of your own domain.

(Note: The semicolons are comment lines. The trailing dot on the hostname, line 2, is mandatory. The IP address in the sample is not a registered address; the one[s] you use for forwarding will be!)

3.03 Edit `named.boot` and add a line for forwarders:

```
;
; type          domain                  source file
directory       /etc/nameserver         ; running directory for named
primary         0.0.127.IN-ADDR.ARPA    db.127.0.0
primary         [your domain name]      db.[domain]
primary         3.1.192.IN-ADDR.ARPA    db.192.1.3
cache           db.cache
forwarders      10.1.3.25
```

3.04 Edit `/etc/resolv.conf` to add the new nameserver:

```
domain [your domain name]
nameserver [IP address of primary]
nameserver [IP address of secondary]
nameserver 10.1.3.25
```

3.05 Restart named:

```
/sbin/init.d/named start
```

3.06 Test name resolution: `nslookup [somewhere].com`

3.07 Revise the `/etc/resolv.conf` file of all clients to include the forwarder's IP address line.

webmaster's toolbox

Apache Modules

by Neil Kohl

Neil Kohl has been the Webmaster for the American College of Physicians – American Society of Internal Medicine for three years. He fell into a career in computers while studying neuropsychology in graduate school.

<nkohl@mail.acponline.org>

I'm the Webmaster for a nonprofit medical membership organization. We have a very large Web site – about 20,000 files – and a very small staff. Like many Webmasters, I rely on a motley collection of tools to do a lot of the daily grunt work of content management, link checking, and statistics. One of the most powerful tools in my toolbox is the server software itself. We use Apache.

Apache is the most popular server on the Internet[1] for a number of reasons: It's fast, it's stable, and it's flexible. Apache's modular architecture makes it easy to modify the server to suit your needs.

For example, the standard distribution comes with three different authentication modules that allow you a choice of back end for storing user information: `mod_auth` for flat files, `mod_auth_db` for Berkeley DB files, and `mod_auth_dbm` for DBM files. If that doesn't fit the bill, you can find modules for authenticating using mSQL, Postgres, Kerberos, Windows NT domain servers, or any external program (`mod_external`). Add cookie authentication with `mod_cookie_auth`. Or you can roll your own. (I've always wanted to write an authentication module that will admit only a quarter of users at random. I'd call it `mod_studio_54`.)

The Apache FTP site has a `/contrib` directory with a number of modules, written by others, available for downloading. In this article I'd like to take a closer look at three modules that come with the standard distribution that can make the Webmaster's job a whole lot easier: `mod_speling` (yes, that's really how it's spelled), `mod_rewrite`, and `mod_usertrack`.

Making Modules Available

Before you can use a module in Apache, it has to be compiled in to the `httpd` binary. Building Apache is beyond the scope of this article; I assume you've already got it built and running.[2]

To see which modules are available, open the file `apache_1.3.x/src/Configuration` (where `apache_1.3.x` is the top level of the source tree for your version of Apache). The modules I describe here are not compiled in by default. If you'd like to use these modules, make sure the following lines in `Configuration` are uncommented:

mod_speling *will correct up to one spelling error in a URL. DOS users who are in the habit of using the ".htm" extension will be able to get to your ".html" pages.*

```
AddModule /modules/standard/mod_speling.o
AddModule /modules/standard/mod_rewrite.o
AddModule /modules/standard/mod_usertrack.o
```

(If you have Apache prior to 1.3, the `Module` command is used to add modules, and the syntax is a little different. Check the documentation.)

Uncomment lines for modules you'd like to use, save `Configuration`, and `make` to build the module and link it into Apache. Copy the new `httpd` binary over your old binary. Restart the Web server and make sure everything is working.

Directives and Context

Each module makes one or more configuration directives available for you to use. The documentation lists the directives that the modules provide and the context in which these directives can be used.

In the examples here, I place the directives in `httpd.conf`, so the entire server is affected. Most of these directives can also be placed within a `VirtualHost` or `Directory` context to limit effects to a subset of the server.

Now let's explore.

mod_speling

The problem: UNIX filesystems are case-sensitive. Most users aren't. Novices are especially fond of the caps-lock key. A caps-lock fan trying to get to the catalog directory at some site might try:

```
HTTP://WWW.SOMESITE.COM/CATALOG/
```

While the hostname part of a URL is case-insensitive, the path and filename are not, and the address above will cause a "File not found" error, assuming that the directory is really `/catalog`.

`mod_speling` makes URLs case-insensitive. The all-caps URL above would return the catalog index page instead of a 404 error. An added bonus is that it will correct up to one spelling error in a URL. DOS users who are in the habit of using the ".htm" extension will be able to get to your ".html" pages.

`mod_speling` provides one directive, `SpellCheck`. Add the following lines to your `httpd.conf`:

```
# turn on mod_speling...
SpellCheck on
```

Restart Apache and turn on caps-lock!

A feature (or drawback, depending upon your point of view) of `mod_speling` is that under certain conditions it will present the user with multiple choices if it can't find a suitable file. For example, if a user types in: http:/www.yoursite.com/index without the .html extension, `mod_speling` will present the following message:

```
Multiple Choices

The document name you requested (/index) could not be found on this
server. However, we found documents with names similar to the one
you requested. Available documents:
/index.html (common basename)

Apache/1.3.6 Server at neilk.acponline.org Port 80
```

On my workstation I have a directory that contains files named `index1.htm` to `index25.htm`. If I accidentally type in http://neilk.acponline.org/brownbag/index53.htm I get the following:

```
Multiple Choices

The document name you requested (/brownbag/index53.htm) could not be
found on this server. However, we found documents with names similar
to the one you requested.

Available documents:

/brownbag/index3.htm (extra character)
/brownbag/index5.htm (extra character)
/brownbag/index13.htm (mistyped character)

Apache/1.3.6 Server at neilk.acponline.org Port 80
```

Spelling correction does consume some resources, which may be a consideration if you have a busy site.

mod_rewrite

A good rule of thumb for Webmasters is that once you put a page up, it should be available at the same URL until the end of time. Even if you track down every last link to a page on your site, do an AltaVista search to find the publicly accessible Web pages that link to it[3] and send email to every Webmaster telling them to update their pages, you still don't know how many intranets and bookmark files have the old address. It looks bad when people see a "File not found" page on your site, even if they followed a link from a page that hasn't been updated since Netscape went public.

While having a policy of keeping URLs active makes for a good user experience, it makes it hard for a Webmaster to do housecleaning. Some of the common problems that crop up:

- You decide to rename a page, and you'd like to bump anyone going to the old address to the new address.
- Registration material for an annual meeting that happened three years ago really makes a site look stale. You'd like to remove these pages and redirect users who attempt to access them to the entry page for this year's annual meeting.
- You have to change a directory name, but the names of the files in the directory remain the same. You'd like to point anyone looking for a file in the old directory to the corresponding file in the new directory.[4]

There are three types of redirection you can do: one-to-one, where all requests for one file are mapped to a different file; many-to-one, where requests for any one of a group of files are mapped to a single file; and many-to-many, where requests for a file in one group of files are mapped to the corresponding file in a second group.

The standard redirection directive, `Redirect`, can only map a URL path to a full URL. `mod_rewrite` allows regular-expression-based URL mapping and redirection. It's incredibly flexible and powerful. As the documentation says, "Welcome to `mod_rewrite`, the Swiss Army Knife of URL manipulation!"

Before you can start creating remappings, you have to turn on the rewrite engine. In `httpd.conf`, add the following line:

```
# turn on mod_rewrite
RewriteEngine On
```

Let's examine each of the different situations I've outlined.

In the first case, let's say that you have to map http://www.somesite.com/foo.htm to http://www.somesite.com/bar.htm. This is an easy one:

As the documentation says, "Welcome to `mod_rewrite`, the Swiss Army Knife of URL manipulation!"

```
# one-to-one redirect: foo.htm to bar.htm
RewriteRule ^/foo.htm /bar.htm
```

Rewrite rules are based on regular expressions. The general form of a rewrite rule is `RewriteRule regexp remapping`. Notice the `^` before `/foo.htm`. This ensures that the rule will be triggered only if the pattern `/foo.htm` is found at the beginning of the path. Here, anytime the page foo.htm in the document root directory is requested, the page /bar.htm will be displayed. Without the `^`, any page called foo.htm will be redirected to bar.htm.

The redirection is invisible to the end user – the URL in the browser's location bar will still be http://www.somesite.com/foo.htm. All of the sleight-of-hand is happening on the server side.

In the second case, assume that you're deleting the `1998meeting` directory and all of the pages in it, and you'd like to redirect requests for any page in the `1998meeting` directory to the main page for this year's meeting, 1999meeting/index.html.

```
# many-to-one mapping
RewriteRule ^/1998meeting.* /1999meeting/index.html [R]
```

Notice the `.*` after `1998meeting` – it's a regular-expression wildcard. Any requests for the directory alone or anything in the directory will match the regex, trigger the `RewriteRule`, and redirect users to the 1999 meeting home page.

In the first example, the user sees the original URL in the browser's location bar even though the server sent a different file. The `[R]` in this case forces a redirect and the "real" URL appears in the end user's browser. If we had left off the `[R]`, the user would see `1998meeting` as the location of the page even though content for 1999 is being displayed. It might look like a mistake.

For the final case, suppose we have a set of pages in `/oldname` and the directory name gets changed to `/newname`. The files in the directory remain unchanged. The task here is to map a request for a specific file in `/oldname` to the corresponding file in `newname`.

```
# many-to-many mapping
RewriteRule ^/oldname(.*) /newname$1 [R]
```

Yep, `mod_rewrite` supports backreferences! In this case, any request for pages in `/oldname` will trigger the `RewriteRule`. The part of the request after `oldname` will be stored in `$1` and will be appended to the new URL after `/newname`. Here again the `[R]` forces a redirect so users will see the new location in their browser.

I've only scratched the surface of what `mod_rewrite` can do. It's an incredibly powerful module that can do conditional URL remapping based on request headers. It can remap requests based on lookups in a file, dbm database, or calls to an external program. You can proxy requests to other Web servers.[5]

mod_usertrack

"How many visitors do you get?" It's one of the most persistent and unanswerable questions a Webmaster faces. You can explain that HTTP is an anonymous stateless protocol and that caching and proxies cloud the picture to the point of hopelessness. People will still ask.

You can count the number of distinct hosts. But how many individual visitors are hiding behind AOL's proxy servers? You could require registration and password-protect your entire site. You'd have a very accurate picture of your number of visitors – declining, most likely, because no one wants another user name and password to remember.

mod_usertrack offers a solution that offers a fairly accurate picture of visitors without requiring a burdensome login. When mod_usertrack is activated, Apache checks each incoming request for a cookie header containing a unique identifier. Cookies received by the server are logged. If the client doesn't pass a cookie in the request, the server generates a new unique identifier and a cookie containing the ID is sent with the server's response.

There is no personal information in the cookie, just the client IP address and a random number. Even though cookies as used here are pretty innocuous, people still get spooked about them. You should think through the privacy implications of being able to uniquely identify users.

To enable mod_usertrack, define a log for your cookie tracking data in httpd.conf:

```
# define the clickstream log format and logfile:
LogFormat "%t %u %{cookie}n %r" clickstream
CustomLog var/log/clickstream clickstream
```

The LogFormat directive shown here will record time, authenticated user name if it's available, the identifier that Apache generates, and the user's request.

Next, decide how long you want cookies to last with the CookieExpires directive. If you don't set an expiration date for cookies, they last for the duration of the user's session, and a count of unique IDs will give you the number of different sessions. If you're interested in how many repeat visitors you get, you'll want to set the expiration to some point in the future:

```
# expiration date for cookie
CookieExpires "3 months"
```

You can specify the expiration in seconds (one number, no label) or as "number time-period [number time-period ..]" where number is a number and time-period is one of years, months, weeks, hours, minutes, or seconds. The expiration time must be enclosed in quotes if it contains spaces.

Then turn user tracking on with this directive:

```
# turn on user tracking...
CookieTracking on | off
```

Restart your server. To prove that it's really sending out cookies, set your browser to ask before accepting cookies and go to your site. You should get a cookie with a value like:

```
Apache=172.168.1.254.12512917405203642
```

You can also look in your browser's cookie file for your site's domain name and the Apache cookie. The clickstream log file will accumulate entries that look like this:

```
[28/Apr/1999:00:00:28 -0400] - 172.24.48.11.3103925272028252 GET
          /index.html HTTP/1.1
[28/Apr/1999:00:00:28 -0400] - 172.188.154.3.2997925271804334 GET
          /outline.htm
[28/Apr/1999:00:00:28 -0400] - 172.188.154.70.3027925271992432 GET
          /search.htm HTTP/1.0
```

Want to know how many different visitors (technically, Web browsers that accepted cookies) you had yesterday? Assuming you used the format above, the answer is:

```
% cut -d " " -f 4 clickstream.| sort | uniq | wc -l
-> 8698
```

Our organization has over 100,000 members. Most of the Web site is publicly accessible, but some pages are available to members only. Any member can register for a user

Even though cookies as used here are pretty innocuous, people still get spooked about them. You should think through the privacy implications of being able to uniquely identify users.

One of the questions I am frequently asked is: What percentage of traffic to public pages is by members? Before we initiated user tracking via `mod_cookie`, *there was no way to answer this question.*

name and password; over 12,000 have. One of the questions I am frequently asked is: What percentage of traffic to public pages is by members? Before we initiated user tracking via `mod_cookie`, there was no way to answer this question.

Members-only pages are protected with HTTP basic authentication. When a member logs in and views a members-only page, the REMOTE_USER name gets recorded in the clickstream log along with the Apache ID. Now we can track member hits in the public area of the Web site by looking for Apache IDs of members who have logged in. It's easy to write a quickie Perl script to estimate the percentage of member hits in the public area:

```
#!/usr/bin/perl -na
# pubhits.pl
# usage: pubhits.pl clickstream-log [ clickstream-log .. ]
if ($F[2] ne '-'){
$memb_ids{$F[3]}++ # record member's Apache ID
} else {
$all_pub_ids{$F[3]}++ # record Apache ID
}
END {
foreach (keys %all_pub_ids){
$t_pub_hits += $all_pub_ids{$_};
$memb_pub_hits += $all_pub_ids{$_} if (defined $memb_ids{$_});
}
printf "Publ hits: %d; Memb publ hits: %d; Pct publ hits by memb:
          %.2f\%\n",
$t_pub_hits, $memb_pub_hits, $memb_pub_hits / $t_pub_hits *100;
}
```

This doesn't give a perfect answer – members who don't log in to the private area are not counted, nor are members who logged in on a different day – but it's a decent estimate.

Combining the Apache ID with the authenticated user name also allows us to estimate the number of browsers per user.

More Information

Taking the time to dig through the Apache documentation and source tree can have significant rewards – your Web server may turn out to be the most useful tool in your toolbox.

Learn more about Apache, modules, and the module API:

Books

Laurie, B., and Laurie, P. *Apache: The Definitive Guide,* 2e. Sebastopol, CA: O'Reilly & Associates, 1999. Compiling and configuring Apache; a tour of the Apache module API.

Stein, L., and MacEachern, D. *Writing Apache Modules in Perl and C.* Sebastopol, CA: O'Reilly & Associates, 1999. Brand new. The Apache module API in detail, with examples of extending Apache using C and mod_perl.

Web

Apache documentation: <http://www.apache.org/docs/index.html>

Apache module registry: <http://modules.apache.org>

Apache user-contributed modules: <ftp://ftp.apache.org/dist/contrib/modules/>

ApacheWeek: <http://www.apacheweek.com/>. Apache news and feature articles that discuss server configuration topics in detail.

Notes

[1] According to Netcraft, <http://www.netcraft.co.uk/>.

[2] Documentation for building Apache is at <http://www.apache.org/docs/install.html>.

[3] This AltaVista query will let you know how many pages have links to your site: `link:www.yoursite.com -url:www.yoursite.com`. The first part (link:) searches for links to your site; the second part excludes links that come from your own site from the results.

[4] True story: We have the text of a book on our Web site, *The Home Care Guide to Cancer.* It was living in a directory called "homecare." One day we received a letter from an intellectual-property law firm informing us that "Homecare" was a registered trademark of another publisher and we had to change the directory name to avoid legal action. They never mentioned the content of the book. Just the directory name.

[5] We have a document management system (DMS) available on our internal network. The DMS has a Web gateway that allows users to search public-policy documents and view them as HTML or PDF. I was asked to make the collection available on our public Web site. This was no easy task, since our Web server is isolated from our internal network in its own DMZ. Both the Web server and the internal network are protected by a firewall.

Moving the DMS was not an option for security and logistical reasons. We didn't have time to implement a CGI solution. After talking this through with our security consultant, we reconfigured the firewall to allow the public Web server HTTP access to the DMS Web gateway. Then I configured Apache to proxy requests for policy documents to the DMS using three `RewriteRules` and one `ProxyPassReverse` directive. One rule in the firewall and four lines of Apache configuration!

managing network security with cfengine, part 1

by Mark Burgess

Mark is associate professor at Oslo College and is the author of cfengine and winner of the best paper award for LISA 1998

<Mark.Burgess@iu.hioslo.no>

Computer security is about protecting the data and availability of an association of hosts. The key words are authentication, privacy, integrity, and trust. To understand computer security we have to understand the interrelationships among all of the hosts and services on our networks as well as the ways in which those hosts can be accessed. Tools that allow this kind of management are complex and usually expensive. A simple free software program that is in widespread use whose functionality is at least equal to commercial packages is cfengine. It runs on nearly every UNIX-like system and on NT with the `cygwin` tools installed.

Software security is about access control and software reliability. No single tool can make computer systems secure. Major blunders have resulted from the belief that a single product (e.g., a firewall) would solve the security problem. For instance, a few years ago a cracker deleted all the user directories from a dial-up login server belonging to a major Norwegian telecommunications company – from the comfort of his Web browser. This was possible, even through a firewall, because the Web server on the host concerned was incorrectly configured. The bottom line is that there is no such thing as a secure operating system, firewall or not. What is required is a persistent mixture of vigilance and adaptability.

Many perceive security as synonymous with network privacy or prevention of network intrusion. Privacy is one aspect of security, but it is not the network that is our special enemy. Many breaches of security happen from within. There is little difference between the dangers of remote access from the network and direct access from a console; privacy is about access control, no matter where the potential intruder might be. If we focus exclusively on network connectivity we ignore a possible threat from internal employees (e.g., the janitor who is a computer expert and holds a grudge; or the mischievous son of the director who was left waiting to play in Mom's office; or perhaps the unthinkable, a disgruntled employee who feel unappreciated). Software security is a vast subject because modern computer systems are complex. It is only exacerbated by the connectivity of the Internet, which allows millions of people to have a go at breaking into networked systems. What this points to is the fact that a secure environment requires a tight control of access control on every host individually, not merely at specific points such as firewalls.

This series of articles is an attempt to illustrate how cfengine can be used to help you automate a level of host integrity on all the hosts of your network. Cfengine is a network-configuration tool with two facets. First, it is a language used to build an "expert system." An expert system describes the way you would like your hosts and network to look and behave. Second, cfengine is a software robot that compares the model you have described with what the world really looks like, and then sets to work correcting any deviations from that picture. In many ways it is analogous to an immune system, neutralizing and repairing damaged parts. Unlike many shell-script packages for system administration, cfengine is a C program, which means that it's light on system resources. It works on a principle of "convergence." Convergence means that each time you run cfengine the system should get closer to the model that you have described

A Word of Warning

It is only proper to state the obvious. You should never trust anyone's advice about configuration or security without running it past your own gray matter first. The examples provided here are just that: examples. They might apply to you as written and they might need to be modified. You should never accept and use an example without thinking carefully and critically first! Also, in any book of recipes or guide to successful living you know that there are simplified answers to complex questions and you should treat them as such. There is no substitute for real understanding.

You can read more about cfengine and obtain it from <http://www.iu.hioslo.no/cfengine>.

until eventually, when the system matches the model, cfengine becomes quiescent, just like an immune system. In the words of one user, your hosts "never get worse." This assumes of course that your model is what you really want.

Model-building using cfengine becomes synonymous with formulating and formalizing a system policy. What makes cfengine a security tool is that security policy is a part of system policy: You cannot have one without the other. You will never have security unless you are in control of your network. Cfengine monitors and indeed repairs hosts with simple, easily controllable actions. From an automation perspective, security is no different from the general day-to-day business of system maintenance; you just need to pay more attention to the details. We cannot speak of "having security" and "not having security." There is always security – it is simply a matter of degree: weak or strong; effective or ineffective.

From an automation perspective, security is no different from the general day-to-day business of system maintenance; you just need to pay more attention to the details.

Automation

Even in the smallest local-area network you will want to build a scheme for automating host configuration and maintenance, because networks have a way of growing from one host into many quite quickly. It is therefore important to build a model that scales. A major reason for using cfengine is scalability. Whether you have one host or a hundred makes little difference. Cfengine is instructed from a central location, but its operation is completely and evenly spread across the network. Each host is responsible for obtaining a copy of the network model from a trusted source and is then responsible for configuring itself without intervention from outside. Unlike some models, cfengine does not have to rely on network communication or remote-object models. We also need integration or "the ability to manage the interrelationships between hosts." It is no good having complete control of one important host and thinking that you are secure. An intruder who can get into any host is almost certain to get into the ones that matter, especially if you are not looking at all of them. Using cfengine is a good way of forcing yourself to formulate a configuration/security policy and then stick to it. Why cfengine? Three reasons:

- It forces a discipline of preparation that focuses you on problems at the right level of detail.
- It provides you with "secure" scalable automation and a common interface to all your hosts.
- It scales to any number of hosts without additional burdens.

The first step in security management is to figure out a security policy. That way, you know what *you* mean by security and what you will do if that security is breached. In many cases you can formulate a large part of your security policy as cfengine code. That makes it formal and accurate, and the robot will do it without requiring any more work on your part.

As an immune system, cfengine will work fine even in a partially connected environment because it makes each host responsible for its own state. It does not rely on network connectivity for remote-method invocations or CORBA-style object requests as does, say, Tivoli. All it needs is an authentic copy of the network-configuration document stored locally on each host. If this is the case, a detached host will not be left unprotected; at worst it might lag behind in its version of the network configuration.

Trust

There are many implicit trust relationships in computer systems. It is crucial to understand them. If you do not, your trust can be exploited by attackers who have thought more carefully than you have.

Cfengine assumes that its input file is secure. Apart from that input file, no part of cfengine accepts or uses any configuration information from outside sources.

For example, any NFS server of users' home directories trusts the root user on the hosts that mount those directories. Some bad accidents are prevented by mapping root to the user "nobody" on remote systems, but this is not security, only convenience. The root user can always use `su` to becomes any user in its password file and access or change any data within those filesystems. The `.rlogin` and `hosts.equiv` files on UNIX machines grant root (or other user) privileges to other hosts without the need for authentication.

If you are collecting software from remote servers, you should make sure that it comes from a machine that you trust, particularly if it includes files that could lead to privileged access to your system. Even checksums are no good unless they also are trustworthy. For example, it would be an extremely foolish idea to copy a binary program such as `/bin/ps` from a host you know nothing about. This program runs with root privileges. If someone were to replace that version of `ps` with a Trojan-horse command, you would have effectively opened your system to attack. Most users trust anonymous FTP servers from which they collect free software. In any remote copy you are setting up an implicit trust relationship. You trust the integrity of the host you are collecting files from, and you trust that it has the same username database with regard to access control. The root user on the collecting host has the same rights to read files as the root user on the server. The same applies to any matched user name.

Why Trust Cfengine?

Cfengine has a very simple trust model. It trusts the integrity of its input file and any data that it explicitly chooses to download. Cfengine places the responsibility on `root@localhost`, not on any outsiders. You can make cfengine destroy your system, just as you can destroy it yourself, but no one else can. As long as you are careful with the input file, you are trusting essentially no one. (I'll qualify this below for remote file copying.)

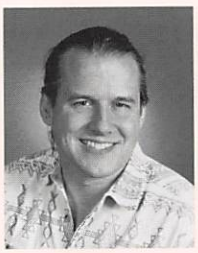
Cfengine assumes that its input file is secure. Apart from that input file, no part of cfengine accepts or uses any configuration information from outside sources. The most one could do from an authenticated network connection is to ask cfengine to carry out (or not) certain parts of its model; thus in the worst-case scenario an outside attacker could spoof cfengine into configuring the host correctly. In short, no one except `root@localhost` can force cfengine to do anything (unless root access to your system has already been compromised by another route). This means that there is a single point of failure. The input file does not even have to be private as long as it is authentic. No one except you can tell cfengine what to do.

There is a catch, though. Cfengine can be used to perform remote file transfer. In remote file transfer one is also forced to trust the integrity of the data received, just as in any remote-copy scheme. Although cfengine works hard to authenticate the identity of the host, once the host's identity is verified it cannot verify the accuracy of unknown data it has been asked to receive. Also, as with all remote file transfers, cfengine could be tricked by a DNS spoofing into connecting to an imposter host. So use the IP addresses of hosts, not their names, if you don't trust your DNS service. In short, these faults are implicit in remote copying. They do not have to do with cfengine itself. This has nothing to do with encryption, as users sometimes believe; encrypted connections do not change these trust relationships – they improve the privacy of the data being transmitted, not its accuracy or trustworthiness.

In the next issue, we will look at the details of cfengine configuration.

on reliability

Security and Reliability



by John Sellens

John Sellens is Director, Network Engineering, at UUNET Canada Inc. in Toronto, after many years as a system administrator. He is also proud to be husband to one and father to two.

<jsellens@uunet.ca>

For most corporations, email is a mission-critical application. It often is the number one communications medium for developers, sales, and customers. However, unsolicited commercial email (UCE or spam) has reached levels at which it is starting to interfere with the effectiveness of email as a communication tool. Separating junk from real email wastes not only network/computing resources but also employee time. More important, many people consider spam to be an invasion of their private mailboxes; arguably the worst aspect of spam is that it demoralizes employees and can even jeopardize their emotional well-being.

Most corporate postmasters have been given the responsibility of dealing with spam. A quick search on the Internet reveals various technical solutions that have been created to help stop spam. One big implementation problem with these anti-spam measures is that they are usually applied on a site-wide basis. For most corporations, some email addresses – such as sales, technical support, and bug reporting – *must* not be blocked. Some of the spammers are our customers; we want their purchase orders to get through but not their spam. We never want to block bug reports from coming in, even if they are from a known spammer.

This article discusses configuration changes that can be made to sendmail rulesets in order to implement an anti-spam filtering policy on a per-user basis. Users can decide if they want to activate anti-spam features and what level of filtering they want.

The Anti-Spam Features of sendmail

Beginning with sendmail 8.8, the `check_*` group of rulesets were added as features. This group of rulesets provids hooks into the SMTP dialog. For the sake of clarity, I'll show the SMTP dialog here:

1. The sending machine issues a `HELO` (or `EHLO`) in which it identifies itself.
2. The sending machine issues a `MAIL FROM` in which it identifies the sender of the message.
3. The sending machine issues a `RCPT TO` in which it identifies the recipient of the message.
4. The sending machine issues a `DATA` to tell the receiving machine it is about to transfer the message.
5. The message is transferred, and the sending machine ends the message with a "." on a line by itself.
6. The receiving machine acknowledges that it got the message, usually by issuing a unique number.

Sendmail 8.8 included the following four check rulesets:

- `check_relay` – this ruleset is called after step 1 in the SMTP dialog above. It is used to prevent unauthorized IPs from connecting to your machine.
- `check_mail` – this ruleset is called after step 2 in the SMTP dialog. It is used to stop mail from known senders.

The main problem with the anti-spam features included with sendmail is that the checks are made too early in the SMTP dialog.

- `check_rcpt` – this ruleset is called after step 3 in the SMTP dialog. It is primarily used to stop relaying (not to be confused with `check_relay` above.) Relaying occurs when an external user sends mail to your server meant for a different external user. They are using your server as a relay for their email. Spammers often do this in order to hide their identity or to take advantage of your resources. Since we know both the sender and recipient at this point, we can decide whether or not the email is relayed.
- `check_compat` – this ruleset is called after step 5 in the SMTP dialog. It can be used to stop delivery of a message after it has been accepted.

Although these `check_*` hooks were provided, it was left to the system administrator to actually develop rules using these hooks. Claus Assmann[1] and Robert Harker[2] maintain a set of effective rules based on these hooks.

When Sendmail 8.9 was released, Eric Allman included some basic anti-spam features that could be configured into sendmail to take advantage of these hooks. By default, Sendmail 8.9 had relaying turned off (implemented in the `check_rcpt` ruleset). Furthermore, you could enable rejection of email based on either a DNS lookup or the results of a database lookup (implemented in the `check_mail ruleset`).

The Problem

The main problem with the anti-spam features included with sendmail is that the checks are made too early in the SMTP dialog. As configured by sendmail, both the DNS and database check are made in `check_mail` (SMTP step 2), after the sender has been identified. If the sender fails the checks, the mail is rejected.

The rejection comes too early because we do not know whom the mail is meant for yet. Also, this means that mail will be bounced regardless of who the recipient was. This is a problem for corporations because there may be some addresses that *must* receive all email. Also, some users may actually want to get spam (true case)!

The Solution

I thought about ways we could block spam for our users while at the same time allowing full access for other addresses. After a little experimenting, I came up with a ruleset that I call "Extended_check_rcpt." Basically, I hold off on the spam checks until the recipient is identified. Then we can check to see if the recipient wants filtering and apply the spam checks as appropriate.

At first I thought about implementing this delayed check by taking advantage of `check_compat`. According to the sendmail book, "Not all situations can be resolved by simply checking the recipient or sender address. Sometimes you will need to make judgments based on pairs of addresses. To handle this situation, V8.8 introduced the `check_compat` rule set."[3] Unfortunately, the problem with `check_compat` is that it is called after the message has been accepted. That means that the sender has already transmitted the message and has closed the connection. If you decide to bounce the message because it fits the spam criteria, your server is then tasked with delivering a bounce message back to the sender. If the sender's address is fake, the bounce messages may back up and clog your mail queue.

Ideally you want to be able to reject a message before it is accepted and the sender has closed the connection. This will shift the burden of delivering a bounce message back to the sending machine. Therefore the best place to apply our spam checks is in `check_rcpt`, after both the sender and recipient are identified but before the message is sent. Fortunately, sendmail stores the sender's address in a macro, and we can use sendmail's delayed macro expansion capabilities to access this value during `check_rcpt`.

Since sendmail supports the use of a database to keep track of spamming addresses, we can create another database to keep track of user preferences. After our modifications are done, the SMTP dialog would look something like this:

1. The sending machine issues a `HELO` (or `EHLO`) in which it identifies itself.
2. The sending machine issues a `MAIL FROM` in which it identifies the sender of the message.
3. The sending machine issues a `RCPT TO` in which it identifies the recipient of the message.
   a. Look into the user database to see if the recipient wants spam filtering.
   b. Apply DNS check if appropriate.
   c. Apply spam database check if appropriate.
   d. Reject the message if step b or c fails, otherwise continue with step 4.
4. The sending machine issues a `DATA` to tell the receiving machine it is about to transfer the message.
5. The message is transferred, and the sending machine ends the message with a "." on a line by itself.
6. The receiving machine acknowledges that it got the message, usually by issuing a unique number.

The new ruleset is called `Extended_check_rcpt` because it is called after the sendmail's `Basic_check_rcpt`, which in turn is called by `check_rcpt`.

`sendmail.cf` Changes to Implement `Extended_check_rcpt`

I'll assume that you already know how to create a `sendmail.cf` file from an `mc` file. You can have other features in your `mc` file, but the two you should have in order to implement `Extended_check_rcpt` are:

```
FEATURE(access_db, dbm -o /etc/mail_access)dnl
FEATURE(accept_unresolvable_domains)dnl
```

Even though the access database checks too early during the SMTP process, it is still a useful feature to enable because the database serves other purposes. First, it is used to enable selective relaying. By listing domains in the access database, you can allow other domains to relay through your site. Second, there may be instances in which you really want to block email from a particular domain, regardless of users' settings. You can use the access database to globally block a particular sender or domain. However, we will not use the access database for "spam stomping," as the config file puts it. We will put our spammers into a different database.

The second feature, "accept unresolvable domains," is necessary because by default sendmail will block email coming from domains that do not exist. As with the access database, this check comes too early. We need to disable this feature so that the DNS check doesn't get included in the config file at `check_mail`. Instead, we will use our own custom code in `check_rcpt` to do the DNS check.

Once the `cf` file has been generated, you will need to hand-edit it to make a few changes.

1. Define the databases. A good place to add the database definitions is after the access database line. My additions are in bold:

```
# Access list database (for spam stomping)
Kaccess dbm -o /etc/mail_access

# Spam database (database of known spammers)
Kspammer dbm -o /etc/spammer

# User opt-in database
Kspamuser dbm -o /etc/spam_user

# Resolve map
Kresolve host -a<OK> -T<TEMP>
```

Explanation: The K configuration options tell sendmail that we will be using two databases (referenced by the names spammers and spamuser). The databases are set up as dbm files, but you can substitute whatever database format you are comfortable with here (db, hash, etc.). The Kresolve map is needed for the DNS check.

2. Modify the current check_rcpt. The check_rcpt ruleset as created by sendmail will look like this (note that the numbers on the left are for reference only and will not appear in the cf file itself):

```
1 SLocal_check_rcpt
2 Scheck_rcpt
3 R$*              $: $1 $| $>"Local_check_rcpt" $1
4 R$* $| $#$*      $#$2
5 R$* $| $*        $@ $>"Basic_check_rcpt" $1
```

Change the check_rcpt ruleset so that it reads:

```
1 Slocal_check_rcpt
2 Scheck_rcpt
3 R$*              $: $1 $| $>"Local_check_rcpt" $1
4 R$* $| $#$*      $#$2
5 R$* $| $*        $: $1 $| $>"Basic_check_rcpt" $1
6 R$* $| OK        $: $1 $| $>"Extended_check_rcpt" $1
7 R$* $| $*        $: $2
```

Explanation: The first thing needed is to change line 5. You'll see that I replaced the $@ with a $:. The reason is that $@ tells sendmail to do the rewrite and exit the rule set. Instead, we want sendmail to do the rewrite and continue to the next line in which we call our customized checks. Also, note that I prepended a $1 $| before the command to execute Basic_check_rcpt. This is needed because Basic_check_rcpt is called with the recipient name but returns with a status condition (OK, err, etc.). We still need to keep track of the recipient so that we can call our customized rulesets with it.

As an example, let's say that the recipient is bond@martini.com. In the original unmodified ruleset, line 5 would be called with the argument bond@martini.com. However, the entire workspace is replaced with the results from Basic_check_rcpt. Assuming that Basic_check_rcpt finds the address acceptable, the workspace would read just "OK." We have lost the original address and cannot use it to call our customized ruleset.

However, with our modified ruleset, line 5 would rewrite our workspace to be:

```
bond@martini.com $| (whatever Basic_check_rcpt returns).
```

Assuming Basic_check_rcpt finds the address to be acceptable, our workspace would now read:

```
bond@martini.com $| OK
```

We can now check for the OK as the second token and call our customized ruleset Extended_check_rcpt with the original address that was saved as the first token. This is what happens with line 6.

3. Insert into your cf file the code for Extended_check_rcpt. This code should be inserted after the last line in Basic_check_rcpt but before the first line of mailer definitions. The source for Extended_check_mail is in the Appendix. You can find an online version of Extended_check_rcpt at <http://www.employees.org/beetle/sendmail.html>.

```
R$*$: [ $1 ]put brackets around it...
R$=w    $@ OK                    ... and see if it is local

# anything else is bogus
R$*     $#error $@ 5.7.1 $: "550 Relaying denied"
```

[Insert code for **Extended_check_rcpt** here]

Mailer Definitions

The code for Extended_check_mail is basically the same DNS check code and access database code as is shipped with sendmail. I have broken out the checks into separate rulesets. I have also added some logic to check the user choice database, which then calls the separate rulesets as necessary. I'll comment on the interesting portions of Extended_check_mail:

```
SExtended_check_rcpt
R$*$: $1 $| $>3 $&f
R$* $| $*$: $2 $| $>3 $1
R$* <@ $* > $| $*       $: <$1 @ $2> $| $3
R$* $| $+ < @ $* > $*   $: $1 $| $2 < @ $3 > $4 $| $2
```

Again, we are making use of the $| token to separate fields in our workspace. The code above basically rearranges our workspace so that it contains:

```
sender  $| recipient              $|     username
```

The username is just the recipient with the @domain chopped off. We will use the username to look into our database to see if this user wants spam filtering.

```
R$* $| $* $| $*  $: $1 $| $2 $| $(spamuser $3 $:<?NOKEY> $)

# No such user in database. Don't do any checks
R$* $| $* $| <?NOKEY>  $@ $2
```

After we get the username as the third field, we pass that into the database lookup. The $:<?NOKEY> tells sendmail to return <?NOKEY> as a default value if no entry is found. If <?NOKEY> is returned, the very next line tells sendmail to return the second field (the recipient) and exit the ruleset.

If the database lookup did return a value, then we will check that value and call My_check_domain (which does the DNS check), or My_check_db (which checks the spammer database), or both. Depending on the results of My_check_db or My_check_domain, we will return either the original recipient or an error and exit.

Creating the Databases

Once you are done making the sendmail configuration file changes, you'll need to create the databases that contain the information. The sendmail distribution comes with a handy tool to do this, called makemap. My sample entries are shown in bold. We'll use these sample entries for testing later.

1. The spam_user database is where your users' preferences are kept. It will be used to tell sendmail if the recipient wants DNS checking, spam database checking, or both.

It's very important to test the new configuration before using it in production. You can test the changes with sendmail's address-test mode (`sendmail -bt`).

To create this database, first create a text file called `spam_user` in this format:

```
Username <choice>
```

where `<choice>` can be `<?DOMAIN>` for DNS checking, `<?DB>` for database checking, or `<?BOTH>` for both checks. After the text file is created, use `makemap` to create the database:

```
tim   <?BOTH>
brad  <?DOMAIN>
john  <?DB>
```

```
% makemap dbm /etc/spam_user < /etc/spam_user
```

Note that the location of the file (in my case `/etc/spam_user`) should correspond with the location you defined with the `Kspamuser` configuration command in step 1. Also, be sure that the database type corresponds.

2. Create the `spammer` database. This is the database where we will keep all of our known spammers. The procedure is essentially the same as creating the `spam_user` database. Make a text file called `spammers` that contains:

```
Address Message
```

where `Address` can be a fully qualified address (somewhere@somewhere.com), an address alone (free.stealth.mailer@), or just a domain (somewhere.com). The message can be either `REJECT` or a customized message (`550 - We don't want spammers here`).

```
free.stealth.mailer@          550 - We don't want spammers here
hot999@aol.com                REJECT
spamrus.com  REJECT
```

```
% makemap dbm /etc/spammers < /etc/spammers
```

3. Create the access database. If you recall, we will not be using the access database for spam stomping but have configured it to take advantage of its other functions. This is the database where we will keep addresses that are allowed to relay and also addresses that will be globally blocked, regardless of how the users have their spam preferences set. Read the sendmail documentation on how to use the access database if you need to enable relaying. For the purposes of this article, I will create an empty access database:

```
% touch /etc/access
% makemap dbm /etc/mail_access < /etc/mail_access
```

Testing the New Configuration

It's very important to test the new configuration before using it in production. You can test the changes with sendmail's address-test mode (`sendmail -bt`).

Before testing, you will need to define the `f` macro value that holds the sender's address:

```
.Df<sender address>
```

Test Case #1 – Tim has enabled both DNS and database checking.

A. Testing mail from someone in the spammer database to Tim. Since Tim has both checks turned on, the system should reject the email.

```
%/usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> .Dfhot999@aol.com
> check_rcpt tim@cisco.com
```

[lots of output deleted]

```
rewrite: ruleset   3 returns:   hot999 < @ aol . com . >
rewrite: ruleset 199 input:     hot999 < @ aol . com . >
rewrite: ruleset 199 returns:   hot999 < @ aol . com . >
rewrite: ruleset 183 returns: $# error $@ 5 . 7 . 1 $: "550 Access
denied"
```

B. Testing mail from a bogus domain to Tim. Since Tim has both checks turned on, the system should reject the email.

```
%/usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> .Dfspam@garbagedomain.com
> check_rcpt tim@cisco.com
```

[lots of output deleted]

```
rewrite: ruleset   3 returns:   spam < @ garbagedomain . com >
rewrite: ruleset 199  input:    spam < @ garbagedomain . com >
rewrite: ruleset 199 returns:   spam < @ garbagedomain . com >
rewrite: ruleset 180 returns:   < PERM >
rewrite: ruleset 181 returns:   < PERM >
rewrite: ruleset 184 returns:   $# error $@ 5 . 1 . 8 $: "501 Sender
domain must exist"
```

C. Testing mail from a valid address not in `spammer` database to Tim. The sender address comes from a valid domain and is not in the spammer database, so the system should allow the email to pass.

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> .Dfuser@cisco.com
> check_rcpt tim@cisco.com
```

[lots of output deleted]

```
rewrite: ruleset   3 returns:   tim < @ cisco . com . >
rewrite: ruleset 199  input:    tim< @ cisco . com . >
rewrite: ruleset 199 returns:   tim < @ cisco . com . >
rewrite: ruleset 179  input:    < cisco . com > < ? > < >
rewrite: ruleset 196  input:    < com > < ? > < >
rewrite: ruleset 196 returns:   < ? > < >
rewrite: ruleset 179 returns:   < ? > < >
rewrite: ruleset 183 returns:   < OK >
```

Test Case #2 – Brad has only enabled the DNS check.

A. Testing mail from someone in the `spammer` database to Brad. Since Brad has only enabled the DNS check, the mail should be accepted.

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> .Dfhot999@aol.com
> check_rcpt brad@cisco.com
```

[lots of output deleted]

```
rewrite: ruleset   3 returns:   hot999 < @ aol . com . >
rewrite: ruleset 199  input:    hot999 < @ aol . com . >
rewrite: ruleset 199 returns:   hot999 < @ aol . com . >
rewrite: ruleset 180 returns:   < OK >
```

B. Testing mail from a bogus domain to Brad. Since Brad has enabled DNS checking, the system should reject the email.

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> .Dfspam@garbagedomain.com
> check_rcpt brad@cisco.com
```

[lots of output deleted]

```
rewrite: ruleset   3 returns:   spam < @ garbagedomain . com >
rewrite: ruleset 199 input:     spam < @ garbagedomain . com >
rewrite: ruleset 199 returns:   spam < @ garbagedomain . com >
rewrite: ruleset 180 returns:   < PERM >
rewrite: ruleset 181 returns:   < PERM >
rewrite: ruleset 184 returns:   $# error $@ 5 . 1 . 8 $: "501 Sender
domain must exist"
```

C. Testing mail from a valid address not in spammer database to Brad. The sender address comes from a valid domain and is not in the spammer database, so the system should allow the email to pass.

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> .Dfuser@cisco.com
> check_rcpt brad@cisco.com
```

[lots of output deleted]

```
rewrite: ruleset   3 returns:   brad < @ cisco . com . >
rewrite: ruleset 199 input:     brad < @ cisco . com . >
rewrite: ruleset 199 returns:   brad < @ cisco . com . >
rewrite: ruleset 180 returns:   < OK >
```

Test Case #3 – the undefined user.

I'll leave it as an exercise for the reader to try John's settings. One more test we should try is for the person who has not set up his or her spam settings. In this case, the system should allow all email to pass through.

A. Mail from an address in the spammer database to Sally. Since Sally is not defined in the spam_user database, the email should be allowed through.

```
% /usr/lib/sendmail -bt
ADDRESS TEST MODE (ruleset 3 NOT automatically invoked)
Enter <ruleset> <address>
> .Dfhot999@aol.com
> check_rcpt sally@cisco.com

rewrite: ruleset   3 returns:   sally < @ cisco . com . >
rewrite: ruleset 184 returns:   sally < @ cisco . com . >
rewrite: ruleset 186 returns:   sally < @ cisco . com . >
```

B. Mail from a bogus domain should be ok too:

```
> .Dfspam@garbagedomain.com
> check_rcpt sally@myhost.com

rewrite: ruleset   3 returns:   sally < @ cisco . com . >
rewrite: ruleset 184 returns:   sally < @ cisco . com . >
rewrite: ruleset 186 returns:   sally < @ cisco . com . >
```

C. Of course, mail from a valid domain that is not in the `spammer` database should be allowed through:

```
> .Dfbailey@cisco.com
> check_rcpt sally@myhost.com

rewrite: ruleset   3 returns:   sally < @ cisco . com . >
rewrite: ruleset 184 returns:   sally < @ cisco . com . >
rewrite: ruleset 186 returns:   sally < @ cisco . com . >
```

Implementation Details

Now that you have tested the sendmail configuration changes, you will have to create some infrastructure to support the new features of `Extended_check_rcpt`. First, the user must have a way to change his or her settings in the `spam_user` database. We have created a Web page so that the user can log in, read about how we fight spam, and change the settings. The CGI then goes and updates the `spam_user` database. If you are a small site, you could probably get away with `rdist`ing a text file and running `makemap` once in a while.

I think it is important to explain to the user the ramifications of turning on the spam checks. In the Web page that we use, we explain that the DNS check may be dangerous and that you should *not* turn it on if it is critical that you get all email. We even have a JavaScript popup that makes the user acknowledge that they have chosen a spam configuration that may cause them to miss email from misconfigured domains.

If the DNS check is too aggressive, then the user can just enable the spam database check. This option is pretty safe to use, since we will only enter addresses from known spammers.

We have set up a mailing list called spam-fighters for our employees to forward spam to. A volunteer team then looks at the email and adds the address to the spammer database if appropriate. Instead of telling users to "just delete the spam," we now tell them to forward it to the spam-fighters alias so that the address can be blocked. Sometimes we can get a report in fast enough that we can block the address before the bulk of the spam comes through. Furthermore, we have created Web pages that show the daily spam statistics, who is in our spammer database, and the number of times we have blocked them. A user can submit a spam report, see that the address gets added to the database, and see the results of that report – that the spammer was blocked from further attempts at connecting to our servers. This is a big PR win for the sysadmin team and a psychological win for the user.

Finally, we try to be proactive about adding spammers into our spam database. We monitor <news.admin.net-abuse.sightings> for spam reports from other sites. We also have set up some fake addresses used by the spam-fighters team. We have publicized these fake addresses by posting to USENET news and using other methods. Since these addresses are not used by anyone inside of the company, any mail going to them is unsolicited and therefore classified as spam.

Overall we have found the checks to be somewhat effective in blocking spam and tremendously effective in fostering the notion that we are actively doing something to curb the spam problem.

Overall we have found the checks to be somewhat effective in blocking spam and tremendously effective in fostering the notion that we are actively doing something to curb the spam problem.

Notes

[1] Claus Assmann's rulesets can be found at <http://www.sendmail.org//%7Eca/ email/check.html>.

[2] Robert Harker's rulesets can be found at <http://www.harker.com>.

[3] Costales, Bryan, and Allman, Eric. *Sendmail.* O'Reilly & Associates, 1997, p. 512.

Appendix: The Extended_check_rcpt Ruleset

```
##########################################################################
### Extended_check_rcpt - called after check_rcpt. Apply DNS and spam db
###     checks on a per-user basis.
##########################################################################
SExtended_check_rcpt

# First expand $&f to get the sender's address.
R$*                             $: $1 $| $>3 $&f
# Apply various rewrites to get workspace into the format:
#  sender $| recipient $| username
R$* $| $*                       $: $2 $| $>3 $1
R$* <@ $* > $| $*               $: <$1 @ $2> $| $3
R$* $| $+ < @ $* > $*           $: $1 $| $2 < @ $3 > $4 $| $2
# Now look into our username to see what kind of checks the user wants
R$* $| $* $| $*                 $: $1 $| $2 $| $(spamuser $3 $:<?NOKEY> $)

# No such user in database. Don't do any checks
R$* $| $* $| <?NOKEY>           $@ $2

# See if user wants domain checking or both and apply check.
R$* $| $* $| <?BOTH>            $: $1 $| $2 $| <?BOTH> $| $>My_check_domain $1
R$* $| $* $| <?DOMAIN>          $: $1 $| $2 $| <?DOMAIN> $| $>My_check_domain $1

# Check results of domain check.
R$* $| $* $| <?BOTH> $| $#error $@ 5.1.8 $: "501 Sender domain must exist"
R$* $| $* $| <?DOMAIN> $$#error $@ 5.1.8 $: "501 Sender domain must exist"
R$* $| $* $| <?BOTH> $| $#error $@ 4.1.8 $: "451 Sender domain must resolve"
R$* $| $* $| <?DOMAIN> $$#error $@ 4.1.8 $: "451 Sender domain must resolve"

# Clear workspace of DNS check results.
R$* $| $* $| $* $| $*           $: $1 $| $2 $| $3

# See if user wants database checking or both and apply check.
R$* $| $* $| <?BOTH>            $: $1 $| $2 $| <?BOTH> $| $>My_check_db $1
R$* $| $* $| <?DB>              $: $1 $| $2 $| <?DB> $| $>My_check_db $1

# Check results. If OK then return original recipient addr and exit.
R$* $| $* $| <?BOTH>  $| <OK>   $@$2
R$* $| $* $| <?DB>    $| <OK>   $@$2

# User didn't want database check, so return original recipient and exit.
R$* $| $* $| $*       $@ $2

# Otherwise, recipient found in spam database, return error from database.
R$* $| $* $| $* $| $*  $@ $4

##########################################################################
### Supporting Rulesets for Extended_check_rcpt. Mostly stolen from
###  stock sendmail package and broken up into separate rulesets.
##########################################################################
SMy_check_db
# check for deferred delivery mode
R$*             $: < ${deliveryMode} > $1
R< d > $*       $@ deferred
R< $* > $*      $: $2

R<>$@           <OK>
R$*             $: <?> $>Parse0 $>3 $          make domain canonical
R<?> $* < @ $+ . > $*  <?> $1 < @ $2 > $3      strip trailing dots
# lookup localpart (user@)
R<$+> $* < @ $+ > $*     $: <USER $(spammer $2@ $: ? $) > <$1> $2 < @ $3 > $4
# no match, try full address (user@domain rest)
R<USER ?> <$+> $* < @ $* $: <USER $(spammer $2@$3$4 $: ? $) > <$1> $2 < @ $3 > $4
# no match, try address (user@domain)
R<USER ?> <$+> $+ < @ $+ $: <USER $(spammer $2@$3 $: ? $) > <$1> $2 < @ $3 > $4
```

```
# no match, try (sub)domain (domain)
R<USER ?> <$+> $* < @ $+ $: $>SpammerLookUpDomain <$3> <$1> <>
# check unqualified user in access database
R<?> $*                 $: <USER $(spammer $1@ $: ? $) > <?> $1

# retransform for further use
R<USER $+> <$+> $*      $: <$1> $3

# check results
R<?> $*                 $@ <OK>
R<OK> $*                $@ <OK>
R<DISCARD> $*           $#discard $: discard
R<REJECT> $*            $#error $@ 5.7.1 $: "550 Access denied"
R<$+> $*                $#error $@ 5.7.1 $: $1 error from access db

SMy_Local_check_domain
SMy_check_domain
R$*             $: $1 $| $>"My_Local_check_domain" $1
R$* $| $#$*     $#$2
R$* $| $*       $@ $>"My_Basic_check_domain" $1

SMy_Basic_check_domain
# check for deferred delivery mode
R$*             $: < ${deliveryMode} > $1
R< d > $*       $@ deferred
R< $* > $*      $: $2
R<>             $@ <OK>
R$*             $: <?> $>Parse0 $>3 $1      make domain canonical
R<?> $* < @ $+ . > $*  <?> $1 < @ $2 > $3   strip trailing dots

# handle non-DNS hostnames (*.bitnet, *.decnet, *.uucp, etc)
R<?> $* < $* $=P > $*   $: <OK> $1 < @ $2 $3 > $4
R<?> $* < @ $+ > $*     $: <? $(resolve $2 $: $2 <PERM> $) > $1 < @ $2 > $3
R<? $* <$->> $* < @ $+>$: <$2> $3 < @ $4 > $5

# handle case of @localhost on address
R<$+> $* < @localhost > $: < ? $&{client_name} > <$1> $2 < @localhost >
R<$+> $* < @localhost.$m$: < ? $&{client_name} > <$1> $2 < @localhost.$m >
R<$+> $* < @localhost.UU$: < ? $&{client_name} > <$1> $2 < @localhost.UUCP >
R<? $=w> <$+> $*        <?> <$2> $3
R<? $+> <$+> $*         $#error $@ 5.5.4 $: "553 Real domain name required"
R<?> <$+> $*            $: <$1> $2

# retransform for further use
R<USER $+> <$+> $*      $: <$1> $3

# handle case of no @domain on address
R<?> $*                 $: < ? $&{client_name} > $1
R<?> $*                 $@ <OK>                     ...local unqualed ok
R<? $+> $*              $#error $@ 5.5.4 $: "553 Domain     ...remote is not
# check results
R<?> $*                 $@ <OK>
R<OK> $*                $@ <OK>
R<TEMP> $*              $@ <TEMP>
R<PERM> $*              $@ <PERM>
R<RELAY> $*             $@ <RELAY>
R<DISCARD> $*           $#discard $: discard

SSpammerLookUpDomain
R<$+> <$+> <$*>         $: < $(spammer $1 $: ? $) > <$1> <$2> <$3>
R<?> <$+.$+> <$+> <$*>  $@ $>LookUpDomain <$2> <$3> <$4>
R<?> <$+> <$+> <$*>     $@ <$2> <$3>
R<$*> <$+> <$+> <$*>    $@ <$1> <$4>

###
### END of Extended_check_rcpt package
###
```

enough SNMP to be dangerous, part 2

by Elizabeth Zwicky

Elizabeth is technical lead of the European Desktop Project at Silicon Graphics. She was a founding member of SAGE and is currently on the USENIX Board of Directors.

<zwicky@greatcircle.com>

This is the second in a series of articles dedicated to teaching typical UNIX system-administrator types – people who can compile public-domain software and have some idea about TCP/IP – how to do UNIX-style hackery with SNMP. It is not an elegant and systematic approach to SNMP, but it should give you enough background to be dangerous.

In part one (*;login:*, December 1998), we looked at `system.sysDescr`. This was an enlightening display of the inability of vendors – even vendors with a strong specialty in SNMP – to get extraordinarily simple parts of the specification correct. However, it's not all that amusing, so let's continue to explore MIB-II and its opportunities for stupid tricks. To look at a chunk of variables at the same time, instead of one, you can use `snmpwalk` where you previously used `snmpget`. To look at the values for everything in the system block, use

```
snmpwalk hostname public system
```

The system block of MIB-II offers the following variables:

```
sysDescr
sysObjectID
sysUpTime
sysContact
sysName
sysLocation
sysServices
```

`sysDescr` we have already seen. `sysObjectID` is defined like this:

> The vendor's authoritative identification of the network management subsystem contained in the entity. This value is allocated within the SMI enterprises subtree (1.3.6.1.4.1) and provides an easy and unambiguous means for determining "what kind of box" is being managed. For example, if vendor "Flintstones, Inc." was assigned the subtree 1.3.6.1.4.1.4242, it could assign the identifier 1.3.6.1.4.1.4242.1.1 to its "Fred Router."

Quick translation: `sysDescr` is useless for most practical purposes. In particular, vendors shove random stuff into it so that two effectively identical machines have different `sysDescr`s (for instance, the `sysDescr` may contain the host name, annoyingly pointless since it occurs later on, or the exact second at which the kernel was compiled). `sysObjectID` will at least identify two boxes as being the same type – and it will also usually point you at a place in the tree where there is more information specific to the particular device. In practice, `sysObjectID` isn't all that useful either, since it generally is much too broad, and most of the truly interesting information is elsewhere. But if you get tired of doing vendor-specific text parsing of `sysDescr`, then `sysObjectID` is a useful hint.

`sysContact` and `sysLocation` are the person responsible for the device and the location of the device, respectively. These are almost always blank. If you have carefully set them on all your devices, you are much too advanced for stupid SNMP tricks.

`sysName` is the system name; the specification says, "By convention this is the fully-qualified domain name." Possibly so; but as you may be beginning to suspect, this is no guarantee. On UNIX machines, it's usually a fully or partially qualified domain name. On Microsoft machines, it's usually the machine's WINS (Microsoft-protocol) name.

On other machines, things are considerably more fluid – it might be the name used in any old protocol, or a name hand-configured by the user, which results in fascinating slippage. (One of our local machines thinks it is named "octwan.cortiallod.sgi.com"; the configurer was trying to type "cortaillod," but actually the domain is named "neu," because "cortaillod" is too hard to type!) Thus, it's usually better to ignore `sysName`, too. Admittedly, only one of the machines I tested managed to produce a protocol error in `sysName`, by returning NULL instead of an empty string.

That leaves `sysServices` and `sysUptime`. `sysServices`, properly interpreted, will tell you which layers of the ISO stack the device is working at. I don't know about you, but I find this less than thrilling. `sysUptime`, on the other hand, is how long the network responder has been up, which is usually very closely related to the length of time the device has been up. This allows you to play amusing, if pointless, games to find out what device on your network has the best uptime.

Assuming you have a list of hosts, and they have SNMP responders started at boot, the following code will settle the uptime argument for once and for all (or at least until people start arguing about which devices count):

```
#!/usr/bin/perl5

use SNMP;

$SEC = 100; $MIN = 60 * $SEC; $HOUR = 60 * $MIN; $DAY = 24 * $HOUR;

while (<>) { # stdin is a list of hosts, one per line
   $hostname = chop;
   if ($sess = new SNMP::Session(DestHost=> "$hostname")){
     $time = $sess->get(["system.sysUptime", "0"]);
   }
   else {
     print "Error: could not bind $hostname\n";
   }

   $uptimes{$time} .= "$hostname ";

}

foreach $uptime (sort numerically keys %uptimes){
   $cutetime = &tformat("$uptime");
   print "$cutetime: $uptimes{$uptime}\n";
}

sub numerically {$b <=> $a;}

sub tformat {
   my($time) = @_;

   my($days) = int($time/$DAY);
   $time = $time - ($days * $DAY);

   my($hours) = int ($time/$HOUR);
   $time = $time - ($hours * $HOUR);

   my($minutes) = int ($time/$MIN);
   $time = $time - ($minutes * $MIN);

   my($seconds) = int ($time/$SEC);
   my($ticks) = $time - ($seconds * $SEC);

   return "$days days, $hours:$minutes:$seconds:$ticks";
}
```

Our local winner was:

```
397 days, 7:47:2:98: mmac1311.neu
```

This is a smart hub, which is the sort of thing that starts people complaining that you really ought to limit the competitors to machines that run actual operating systems.

sysServices, properly interpreted, will tell you which layers of the ISO stack the device is working at. I don't know about you, but I find this less than thrilling

How do I know it's a smart hub? Not from `system.sysDescr.0`, which is "IRM2 SNMP Version 1.00.10", which means nothing to me. However, `system.sysObjectID.0` is "enterprises.52". Being an enterprising person myself, I went to <ftp://ftp.isi.edu/in-notes/iana/assignments/enterprise-numbers>, which told me that enterprise 52 is Cabletron. Doing a search on Cabletron's site showed me that the only place on the entire site that mentions "IRM2" is in fact the documentation for their SNMP management application. This starts out "The IRM2 (Intelligent Repeater Module) provides intelligence to a Cabletron Systems Multi Media Access Center (MMAC) and serves as an IEEE 802.3 repeater for the hub." This is one of the few situations where `sysServices` might have been useful; knowing that I was looking at a device that operated only at level 1 would have been a tip-off. However, the IRM2 doesn't provide `sysServices`. You also might have hoped for a `sysObjectID` more revealing than the top level node for Cabletron, but the IRM2 doesn't implement anything except the bare essentials of MIB-II.

Next: Getting practical – SNMP diagnostics for network problems.

optimal peripheral access using pipe-based double-buffering

by Diomidis Spinellis

Diomidis Spinellis holds a Ph.D. in computer science. Currently he is leading software development at SENA S.A. His research interests include information security, software engineering, and programming languages.

<dds@sena.gr>

When working with slow peripherals such as tape drives and CD-R recorders, careful tuning of the access patterns and process architecture can result in substantial increases in performance. One of the responsibilities of an operating system is to optimize the utilization of a computer system by the appropriate scheduling of tasks. Thus, when a process waits for a peripheral to complete an operation, the OS can schedule other processes that are waiting to run. Although scheduling happens behind the scenes of normal system operation and is transparent to the user, in some cases one can optimize execution time by carefully arranging interdependent tasks. With ever-increasing processor speeds and with peripheral access times relatively stagnant, such optimizations can deliver impressive results. I will describe and explain how I reduced the overhead of tape access by almost 30 percent by simply adding two `dd(1)` commands to our backup pipeline.

The Problem

The problem I wanted to solve was to minimize the actual wall-clock time taken by our backup process, which currently needs eight hours for a full backup. I thought I could achieve this by making our drive *stream*, and by increasing the parallelism of the backup process. Our LAN consists of Windows 95/98/NT and UNIX machines. We perform our backups on a machine running Linux (2.0.36). At backup time, the Linux machine mounts the disks of all Windows hosts using the SMB filesystem and runs a `tar(1)` – based backup process.

The process is responsible for:

- administering incremental and full backups
- maintaining backup quota
- measuring the backup set size
- comparing the files after backup
- logging the backup results

The heart of the process is essentially the following command:

```
tar czf /dev/tape -b 1000 /smb
```

The command creates a `tar` file in `/dev/tape` from the contents of `smb` using `gzip` compression and a blocking factor of 1000. The blocking factor is supposed to make tar write to the tape device in chunks of 1000 blocks (i.e., of 0.5MB).

A golden rule of performance optimization is to profile the process to be optimized.

Accessing the tape in large blocks is mandatory for some devices, and – according to some man pages for `tar` – can be more efficient than access in smaller chunks. Since tape drives typically read or write data at a fixed rate, if that rate cannot be maintained on the rest of the system the drive has to read or write some data and then stop and backtrack to a point where the next block can be written. When this phenomenon occurs, the performance of the drive drastically deteriorates, and the drive is no longer said to be streaming. Writing in large blocks can help the drive to stream.

Our drive consistently refused to stream. Increasing the size of the blocking factor had absolutely no effect. Going through the source code of tar we are using (GNU tar 1.11.8 with a small local modification to handle the lack of stable inode numbers in SMB filesystems), I noticed that when compression is specified, tar ignores the blocking factor. That explained the lack of streaming and prompted me to examine the matter further.

Measurements and the Solution

A golden rule of performance optimization is to profile the process to be optimized. I therefore first measured the time it takes for tar and gzip to process a medium-sized directory:

```
% time sh -c 'tar cf - /usr/src/linux | gzip -c >/dev/null'
375.40user 39.38system 7:02.18elapsed 98%CPU
```

This measurement establishes our performance bottom line. No matter how much we improve tape access, we will never be able to make the process faster than seven minutes. The next step was to measure the overhead of writing to tape:

```
% time sh -c 'tar cf - /usr/src/linux | gzip -c >/dev/tape'
389.81user 37.34system 10:19.96elapsed 68%CPU
```

So we now know that writing to the tape increases the execution time by about three minutes, or 47 percent. It is this increase that we want to fight. Since our full backup takes about eight hours, optimizing tape access could result in significant time savings. The measurement also tells us that the CPU is idle about 30 percent of the time. Our task mix is therefore I/O bound, and we have CPU power to spare in order to optimize it. As a first try, I piped the `gzip` output through `dd(1)` specifying a large (0.5MB) output block size. This causes `dd` to accumulate data from its input until it has 0.5MB of data available and then write that data in one large chunk to the tape:

```
% time sh -c 'tar cf - /usr/src/linux | gzip -c |
  dd obs=512k of=/dev/tape'
27979+1 records in
27+1 records out
392.65user 41.55system 11:28.28elapsed 63%CPU
```

This time I could hear and see the tape drive stream; a beautiful sound and sight. Embarrassingly, the execution time *increased* by more than one minute (11 percent), while CPU utilization decreased. Streaming the tape drive did not appear to help. An analysis of the situation was now in order.

Apparently, in the system I am using, write commands to the tape device are processed synchronously; that is, the process issuing the write system call is suspended until the data is written to the device. In the first case involving the tape drive, while data was being written by `gzip` to the tape drive (and `gzip` was waiting), `tar` could continue to process some files writing its output to the pipe buffer connecting the two processes. In the second case where `dd` was added, `dd` took a longer time to write the 0.5MB block to the tape drive. During that time, the pipe buffer between `gzip` and `dd` got full, thereby

blocking `gzip`. That, in turn, made the pipe buffer between `tar` and `gzip` fill up, blocking `tar` as well. The explanation of the timing anomaly made me realize the problem's root and think about a possible solution.

In order to optimize the backup process I would need a way to prevent the pipeline stalls while data was being written to the tape drive. The addition of another `dd` process after `gzip` could provide this buffer:

```
% tar cf - /usr/src/linux | gzip -c | \
 dd obs=512k | dd obs=512k of=/dev/tape
```

In the pipeline above, `gzip` writes data to the first `dd` process. As soon as this process accumulates 0.5MB, the data is written to the second `dd` process using a single `write` call. As soon as the second `dd` process accumulates 0.5MB (i.e., immediately after the first process writes it), it will issue a `write` call to send the data to the tape drive. While data is being written to the tape, the second `dd` process is blocked; however, the first one is still running, since it has written its block to the second process and is now again accumulating the next block to write. Therefore `tar` and `gzip` can continue gathering and compressing data at the same time that data is being written to the tape drive. Thus, the two `dd` processes effectively implement a double-buffering mechanism. The time measurements prove the validity of this approach:

```
% time sh -c 'tar cf - /usr/src/linux | gzip -c |
  dd obs=512k | dd obs=512k of=/dev/tape
27979+1 records in
27+1 records out
27979+1 records in
27+1 records out
364.34user 48.57system 9:23.22elapsed 73%CPU
```

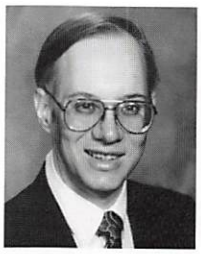
Our change resulted in a 28 percent decrease in the tape overhead, giving a 10 percent decrease in overall execution time. Depending on the performance differences between the CPU and the peripheral in question, the execution time can decrease even more.

Our change resulted in a 28 percent decrease in the tape overhead, giving a 10 percent decrease in overall execution time.

Conclusions

As I mentioned at the beginning, one of the responsibilities of an OS is to optimize the utilization of a computer system by the appropriate scheduling of tasks. In our initial case, while the system was waiting for the tape drive to write the data, it could indeed run other unrelated processes such as `sendmail` or `httpd`. However, the operation of the processes in the backup pipeline could not be parallelized beyond the limits imposed by the small buffer provided by the implementation of pipes under UNIX. Soon after one process in the pipeline blocked, the whole pipeline was stalled. The addition of another `dd` process before the process writing to the tape provided extra breathing space to the pipeline and increased the possibility for parallel operation. The same approach can be used in all cases where a pipeline consists of a mixture of I/O and CPU-bound tasks.

the tclsh spot



by Clif Flynt

Clif Flynt has been a professional programmer for almost twenty years, and a Tcl advocate for the past four. He consults on Tcl/Tk and Internet applications.

<clif@cflynt.com>

There's probably a 12-step program to help those of us who still prefer to use a text-based mail reader in this day of GUIs and multimedia MIME attachments. Some of us still get by quite well with these curses-based dinosaurs. If our mail reader supports hooks for Bellcore's metamail package and the default set of viewers, we can read anything that the GUI-based readers can send.

Except HTML. HTML was developed well after metamail, and while there are simple hooks in the `mailcap` file to invoke Netscape, I don't want to start up a full browser just to read one email message with an HTML attachment.

So, after grousing about this enough times, I put together a simple little HTML text viewer. The program is pretty simple: a text widget, a scrollbar, a quit button, and a call to the HTML-rendering package.

```
#!/usr/local/bin/wish
source "htmllib.tcl"
# build and display the button, text widget, and scrollbar
set b [button .b -text "quit" -command "exit"]
set t [text .t -yscrollcommand ".sy set" ]
set s [scrollbar .sy -orient vertical -command "$t yview"]
grid $t -row 0 -column 0
grid $b -row 1 -column 0
grid $s -row 0 -column 1 -sticky ns
# Read from a file if there's one on the command line, else stdin
if {$argc >= 1} {
   set infl [open [lindex $argv [expr $argc - 1]]]
} else {
   set infl stdin
}
# Initialize the HTML display package.
HMinit_win $t
# Read the input, and display it.
HMparse_html [read $infl ] "HMrender $t"
```

Rather obviously, the main functionality in this program is in `htmllib.tcl`. The command `source "htmllib.tcl"` merges the HTML library into the HTML viewer at runtime. The `source` command loads a script into a Tcl program and evaluates the commands in that script before evaluating the next line of the original script.

Syntax: `source` *`fileName`*

This is similar to the C language `#include` or the C-Shell `source` command. The `source` command is the easiest way to split your package into multiple source-code modules. (There are other ways, and I'll get to them in future articles.)

Steve Uhler wrote the `htmllib.tcl` package while he was with the Tcl group at Sun Labs. Since then, Steve Ball and others have used and modified it. This package shows some tricks you can play with the `proc` and `eval` commands, and demonstrates the power of the Tcl text widget.

Like most programming languages, Tcl allows programmers to split pieces of functionality into subroutines to create modular code. The Tcl subroutine is sometimes called a procedure and sometimes referred to by the name of the Tcl command that creates a procedure, `proc`.

The syntax for the `proc` command is:

Syntax: `proc` *`name args body`*

The `args` and `body` parameters are generally grouped with braces when you create a procedure in your application:

```
proc mySubroutine {arg1 arg2} {
   puts "arg1 is: $arg1"
   puts "arg2 is: $arg2"
   return "DONE"
}
```

In Tcl every line starts with a command. So, Tcl doesn't declare procedures the way we're used to with C or FORTRAN. The word `proc` is a command, not a declaration. Despite the fact that it *looks* a lot like a declaration that `mySubroutine` is a procedure, what's actually happening is that the `proc` command is creating a new procedure and adding it to the procedure hash table.

The Tcl interpreter will not only evaluate scripts you've created with an editor; it will also evaluate lines that are created at runtime by the program being evaluated.

The command that will evaluate a line of text is `eval`.

Syntax: `eval` *`string`*

The string is any valid Tcl command.

The `eval` command is the trick that makes the HTML parsing code in `htmllib.tcl` elegantly simple. The `htmllib.tcl` package uses a set of regular expressions to convert an HTML page from a list of tags and strings into a set of Tcl commands. The `htmllib` package uses `eval` to evaluate the Tcl procedures that render the HTML page into displayed text.

Tcl regular-expression commands are rich enough to fill more than a single "Tclsh Spot" article. In the `htmllib.tcl` package, they are used to convert this text:

```
This is <B>bold</B> and <I>italics</I>.
```

to these Tcl commands:

```
HMrender .t {P} {} {} {This is }
HMrender .t {B} {} {} {bold}
HMrender .t {B} {/} {} { and }
HMrender .t {I} {} {} {italics}
HMrender .t {I} {/} {} {.}
```

Within the `HMrender` procedure, the HTML tag information is massaged into new procedure names such as `HMtag_b` and `HMtag_\b`. These procedures are then evaluated to insert strings into the text widget with the appropriate formatting – which finally brings us to the `text` widget and some of the things you can do with it. My previous article (*;login:*, June 1999) showed how to create a `text` widget and insert text. Now, let's look at what we can do to control the appearance of that text.

The contents of a `text` widget are addressed by their index. An index is a line and character position in the format `line.character`. To match other UNIX utilities, the line numbers start at 1, and the character positions start at 0. The index 1.0 represents the first character in a `text` widget, and the index `2.3` is the fourth character on the second line.

The `text` widget allows scripts to tag areas of text and then define how to display text within that area. A tag consists of a reference name for the tag, and the start and end indices of the area associated with that tag.

The `htmllib.tcl` package uses a set of regular expressions to convert an HTML page from a list of tags and strings into a set of Tcl commands.

A tag is added to a text widget with the *textName* tag add command.

Syntax: *textName* tag add *tagName startIndex1 ?end1? ?start2? ... ?endN?*

textName	The text widget that will contain the tag.
tag add	Add a tag at the defined index points.
startIndex ?end?	The tag will be attached to the character at *startIndex* and will contain all characters up to, but not including, the character at *end*. If the *end* is less than the *startIndex*, or if the *startIndex* does not refer to any character in the text widget, then no characters are tagged.

Text within a tagged area is manipulated with the *textName* tag configure command.

Syntax: *textName* tag configure *tagName -parameter value*

textName	The name of the text widget.
tag configure	Configure the text within a tagged area.
tagName	The name of the tag that defines the range of characters that will be configured.
-parameter	The parameter to be defined for this area. The parameters include: -font — Set the font for this text. -foreground — Set the foreground color for this text. -background — Set the background color for this text.
value	The value to use for this parameter.

As a simple example, this short script generates this display:

```
text .t -height 2 -width 20 -font {times 16 normal}
pack .t
.t insert end "This is bold text"
.t tag configure loud -font {times 16 bold}
.t tag add loud 1.8 1.12
```

This is **bold** text

That explains how the text widget can render different styles of text, but how does it handle hypertext references?

The text and canvas widgets support a bind command that lets you bind a script to an event. Whenever an event (a button press or RESIZE event, for instance) occurs, the defined script will be evaluated. The script may be multiple lines of commands, or a single call to a Tcl procedure.

The text widget supports binding actions to events that happen to tagged areas.

Syntax: *textName* tag bind *tagName ?eventType? ?script?*

textName	The name of the text widget.
tag bind	Bind an action to an event occurring on the tagged section of text, or return the script to be evaluated when an event occurs.

tagName	The name of the tag that defines the range of characters that will accept an event.
?eventType?	If the *eventType* field is set, this defines the event that will trigger this action. Tk supports physical events (button clicks, key presses, etc.), window manager events (enter/leave window, etc.), and virtual events (combinations of events grouped by the user).
script	The script to evaluate when this event occurs.

Adding this line to the previous example would turn the word "bold" red when a user clicks on it.

```
.t tag bind loud {.t tag configure loud -foreground red}
```

In the TclTutor package, I use this technique to bring up TkMan (or the Windows help viewer) when a user clicks on a word.

In the `htmllib.tcl` package, the `bind` command is used to invoke a procedure to handle hypertext references.

In the `htmllib.tcl` package, the `bind` command is used to invoke a procedure to handle hypertext references.

Finally, the HTML viewer uses a `scrollbar` to shift the displayed section of the `text` widget. The Tcl `scrollbar` widget sends commands to a target widget requesting that widget to modify its state to match the `scrollbar`. The `scrollbar` receives commands from the target widget to modify its appearance when the target's state changes.

Using the Tcl `scrollbar` to control a `text` or `canvas` widget is fairly simple:

1. Your script creates a scrollbar and a widget to be controlled by the scrollbar.
2. The two widgets are linked with the `-command` and `-yscrollcommand` (or `-xscrollcommand`) options to exchange information when their state changes.

After those steps are complete, everything else is automatic.

Syntax: `scrollbar` *`scrollbarName ?options?`*

`scrollbar`	Create a scrollbar widget.
`scrollbarName`	The name for this scrollbar
`options`	This widget supports several options. The `-command` option is required.
`-command` *`"procName ?args?"`*	This defines the command to invoke when the state of the `scrollbar` changes. Arguments that define the changed state will be appended to the arguments defined in this option.
`-orient` *`direction`*	Defines the orientation for the scrollbar. The *`direction`* may be `horizontal` or `vertical`. Defaults to `vertical`.

In the HTML viewer, the code `-command "$t yview"` tells the `scrollbar` to invoke the `text` widget's `yview` subcommand with new parameters when a user modifies the `scrollbar`. The `text` widget option `-yscrollcommand ".sy set"` causes the `text` widget to send information to the `scrollbar` whenever its state changes.

That describes some of the details hiding inside the trivial little 20-line HTML viewing script. All you need to do is copy that file into your `/usr/local/bin` directory and add

the following line to your `mailcap` file and you can view HTML messages from elm, pine, or your favorite curses-based mail reader (assuming you run your curses-based mail reader from an X-Term window).

```
text/html; /usr/local/bin/htmlview.tcl
```

Steve Uhler's original `html_library.tcl` file is available at <ftp://ftp.scriptics.com/pub/tcl/misc/html_library-0.3.tar.gz.>. The `plume` Tcl/Tk browser with Steve Ball's modifications to this package can be found at <http://tcktk.anu.edu.au/> or <http://www.neosoft.com/tcl/ftparchive/sorted/apps/plume-0.6.2/>. A version of the `htmlview.tcl` program with `htmllib.tcl` included is available from <http://www.cflynt.com>.

source code UNIX

What's Your Data Worth?

by Bob Gray

Bob Gray is co-founder of Boulder Labs, a software consulting company. Designing architectures for performance has been his focus ever since he built an image processor system on UNIX in the late 1970s. He has a Ph.D. in computer science from the University of Colorado.

<bob@boulderlabs.com>

Thanks to Michael Durian and Steve Gaede for suggestions and comments.

The title could also be phrased, "What's your time worth if you have to re-create lost data?" Last issue we took steps to protect computers from Internet invaders. Now let's consider the threats of equipment failures, user mistakes, and natural disasters. What can we do to minimize the damage from these events? "Make copies" is the answer and the topic of this article.

Backups are redundant copies of computer information that enable reconstruction of the destroyed primary version. The typical backup medium is magnetic tape; alternatives include CD-ROMs, FLASH cards, "floppies," and other separate hard disks. There are many factors to consider when formulating a backup policy and I have written a section on each of these: tradeoffs of data protection, pros and cons of various media, and backup mechanisms and strategies.

Tradeoffs

The time required to create backups periodically has to be balanced against the risk of not having your data and system available. When you have a loss, you can always re-create the system – but it might take weeks. What is the cost of not having that information for a period? Without your system running, how will you process new transactions and information?

I hate typing a document twice – so I'll take measures to reduce the chance of it happening. Consequently, this article will be backed up many times before it is finally published. (For this small amount of text I'll partly rely on emailing the document to myself on another machine.) As the deadline gets closer, I'll become more cautious because I have more to lose and less time to recreate the document.

In the last few years, I've noticed two disturbing patterns: Too many people naively trust their disks, and huge cheap disks are unmanageable for many. It's no surprise that the proletarian computer owners have no idea of the intricacies of a modern disk. They've never seen a disk platter that's sustained a head crash. Their technology models are mostly things like automotive component failures that cause inconvenience. But a

more suitable model is an airplane with a structural failure or a boat with a sizable hole – it's much more than just an inconvenience.

Over 25 years, I've seen no dramatic increase in the reliability of disks. Yes, they've become much faster, their capacity has grown by orders of magnitude, and the price per byte has fallen by orders of magnitude. But the invariant is that a given disk still has a significant probability of failure during its lifetime. I'm delighted that $150 buys me 9GB of storage, but I'll trust my *;login:* article on it about as much as I'd trust the article on a 1970s DEC RK05 disk that held 1MB for thousands of dollars.

Change your mindset – disks do fail.

So change your mindset – disks do fail. I've seen several PC IDE disks fail in the last few months. One of my past projects involved storing movies on disks. We saw plenty of dead-on-arrival disks from the factory and large numbers of "infant mortality" disks (ones that fail early in their lifetime). Disks that were put into production stabilized to failure rates of just several per month in a thousand-drive video server. (Of course, RAID was used to keep the video server running while the bad drive was hot-swapped for a good one.) I suggest you think of today's disks as people thought of car tires in the early 1900s – you know you are going to have a flat, so carry a spare and know how to change it. All disks wear out and fail.

The other scary trend is the proliferation of multi-gigabyte disks in personal computers. It's one thing to have tens of gigabytes on a shared server maintained by a staff of administrators, but when every Tom, Dick, and Harry gets 10GB on a personal computer, there is going to be disaster sooner or later. It's too easy to collect software and data. Never before has humanity had the ability to "pack rat" stuff to this degree. At least stamp collectors have to physically exchange albums and provide shelf space for the collection. But PC owners have their disks connected to an infinite supply of bits – the Internet – where a mouse click turns on the faucet.

Trusting huge cheap disks by itself isn't a problem, but without backups it is. Most PC owners have no way to segregate their data. What's highly valuable and hard to re-create? What's reproducible by taking the time to find the distribution CD-ROMs and reload? What material came from the Internet? Are all of the locations recorded? Do those locations still serve the data? How much transport time will be required? What is plain junk? It isn't practical to handle gigabytes of materials with "point-and-click" techniques.

Of the people you know with PCs, how many of them have tape drives? When was the last time you saw a nationally advertised PC system that included a tape drive? Our industry is doing a disservice to the masses – encouraging them to accumulate programs, games, and data without providing a mechanism for backup. When the disk fails the user gets totally screwed. They buy another disk, spend weeks patching their environment back together, and use floppies more often. Some will buy a 100MB Zip, or even a 1GB Jaz drive, but these are rarely used properly (definition of *properly*: can a failed hard drive be quickly re-created?). As mainstream PC users migrate to Source Code UNIX, we need to help them adopt better practices.

Media

"Best" tape media discussions are endless. I offer some ideas in <http://www.boulderlabs.com/hardware.html>. Look at drive cost, media cost, capacity, and transfer rate. Be sure the transfer rate is acceptable to you – many drives do less than 1GB/hour, so we're talking about more than overnight for 10GB. You'll want a drive with capacity significantly larger than that of your disk(s). Who will be around at 4:00 am to change the 8GB tape when backing up a 10GB drive? Stay with brand-name drives and mainstream formats such as 4mm DDS, 8mm, or DLT and you'll be fine.

Incremental dumps stored on disk create a sort of "poor man's RAID."

After you create the backups there are two more *extremely* important steps:

- Verify that the tape can be read on a different drive.
- Store the tape off-site.

The verification step doesn't have to be performed for every tape, but do it periodically to ensure that your tape-writing drive isn't misaligned and producing garbage. In a business environment, it's best to give the responsibility of creating backups to one person and the responsibility of verifying the backups to another. The second person can also be responsible for off-site storage. The basement of the same office is not a remote location. A number of firms affected by the World Trade Center bombing learned this lesson the hard way.

Tapes aren't the only good media for backups. Writable CD-ROMs have a role. The media is cheap and you end up with a random-access copy. Most CD systems deal only with 650MB of data, but the new, multi-gigabyte DVD equipment is gaining acceptance.

I've used an additional hard disk for backups in many environments during my career. Sometimes it makes sense to create an image copy of your primary disk. This is fast and you end up with something immediately bootable. (It's also handy for backing up foreign operating systems.) SCSI controllers do most of the work of remapping damaged disk sectors, so the old problem of bad blocks for image copies isn't too serious. Under FreeBSD,

```
dd if=/dev/rsd0 of=/dev/rsd1 bs=64b
```

takes just a few minutes to transfer 1GB, and by switching SCSI ID numbers you can make the copy immediately become the new primary disk. This technique requires either identically sized disks or a good understanding of disk labels, partitions, and what is going on. Write me if you want more details.

Another way to use an additional disk is to create a complete dump onto tape and to put the newly changed files on the extra disk. (The dump program makes this easy.) The process can be automated with scripts and you don't have to be tending the tape drive on a daily basis. Further, your incremental changes are immediately accessible online. But make sure that you take some copies off-site. You can extend this practice by dumping over a network, maybe even to an off-site machine.

Incremental dumps stored on disk create a sort of "poor man's RAID." The next improvement might be software disk mirroring. You have the operating system write the data twice to different drives. Then a single drive failure isn't a big problem. Preferably, you would have real RAID controllers with optional hot-swappable drives. These solutions are not expensive, and you get protection and in many cases higher aggregate disk bandwidth. In an NFS environment, a Network Appliance box is the Cadillac of servers. It protects your data with RAID and keeps a number of recent versions of your files online.

For other ideas and opinions see <http://www.freebsd.org/handbook/backups.html> and the Linux HOWTOs at <http://metalab.unc.edu/mdw/linux.html>.

Strategies

I highly recommend that you organize your information in a way that facilitates backups. If possible, separate the relatively static, unchanging material from your core development. UNIX partitions enable you to implement different backup policies for different material. It's common to find these partitions on UNIX systems: `root`, `tmp`, `var`, and `users`.

Once configured, a root partition is mostly static and won't require frequent backups. Monthly might be appropriate. The `/tmp` filesystem by its nature doesn't require a backup. The `/var` filesystem has highly transient files such as mail messages and print spool files; you had better not lose someone's queued mail!

The essence of a company's development is probably in the user filesystem. For software-development shops and content developers, I believe daily backups are appropriate. The equation to balance is the cost of backups versus the cost of loss. The latter is a function of the risk of loss, the amount of work at risk, the cost of the work, and the time required to reproduce the work. As a project approaches a delivery deadline, the cost of loss rises because there is less time to recover and still make the delivery. Therefore the value of frequent backups increases significantly.

I highly recommend that you organize your information in a way that facilitates backups.

The UNIX `dump` program is the mainstay of data protection. It is the most reliable program across UNIX systems for making a perfect copy of your filesystem. (Most UNIX `dump` programs correctly handle the fringe filesystem features like sparse files, long funny names, and special files.) You start by creating a "level 0" dump; it has everything needed to re-create your environment on a fresh disk. The level 0 will take a while. (I require about four hours to back up a couple of gigabytes onto some DAT tapes.) Successively higher dump levels save files that are newer than when a lower dump level was taken. Here is a simple example

Weekend: Level 0
Monday: Level 1
Tuesday: Level 2
Wednesday: Level 3
Thursday: Level 4
Friday: Level 5

The weekend dump will take a lot of time and tape. The weekday dumps will capture only what has changed since the previous day. These are called "incremental" dumps – mine take only a few minutes. The small disadvantage of incremental dumps is that to re-create a given state of the system, you must have that day's backup plus the sequence of lower-level backups. The strategy below depends only on the weekend level 0 dump, and the incremental dump of interest. As the week progresses, the dumps will take longer and require more space.

Weekend: Level 0
Monday: Level 1
Tuesday: Level 1
Wednesday: Level 1
Thursday: Level 1
Friday: Level 1

Here is a suggestion from SGI's IRIX dump manual:

> To guard against data loss as a result of a media failure (a rare but possible occurrence), it is a good idea to capture active files on (at least) two sets of dump volumes. Keep unnecessary duplication of files to a minimum to save both operator time and media storage. A third consideration is the ease with which a particular backed-up version of a file can be located and restored. The following four-week schedule offers a reasonable trade-off among these goals. Although the Tuesday-through-Friday incrementals contain extra copies of files from Monday, this scheme assures that any file modified during the

	Sun	Mon	Tue	Wed	Thu	Fri
Week 1:	Full	5	5	5	5	3
Week 2:		5	5	5	5	3
Week 3:		5	5	5	5	3
Week 4:		5	5	5	5	3

One of the reasons we don't hear of many disk-failure catastrophes is that competent system administrators are quietly doing their jobs.

week can be recovered from the previous day's incremental dump.

Many UNIX variants recommend the modified Tower of Hanoi algorithm using the dump-level sequence 3 2 5 4 7 6 9 8 9 9 . . . From the FreeBSD man page:

> Each week, a level 1 dump is taken, and the daily Hanoi sequence repeats beginning with 3. For weekly dumps, another fixed set of tapes per dumped file system is used, also on a cyclical basis. After several months or so, the daily and weekly tapes should get rotated out of the dump cycle and fresh tapes brought in.

Tar is the fundamental and universal "tape archiving" program in the UNIX world. It copies hierarchies to and from tape, disk, or pipelines. Tar is better than dump for creating backups that may need to be restored onto a different flavor of UNIX. The dump incompatibilities are due to the fact that it intimately deals with the filesystem, whereas tar just uses the filesystem services. But then tar doesn't save all the filesystem information such as last access time or creation time, which may be important. Frequently, tar is used with a compression program. So common is the command `tar cf - . | gzip > XYZ.tz` that GNU tar has compression built in as an option: `tar czf XYZ.tz`. All over the Internet, source code and documentation is provided in tar-gzip files. (In the Windows world, pkzip or winzip provide similar capabilities, but the compression is not as good.)

Pax is the superset archiver. It can generate and handle IEEE Std1003.2 ("POSIX.2") TAR and CPIO formats plus several of the prestandard versions. Pax has more flexibility than tar. It has a built-in mechanism for selecting files on the basis of a range of timestamps. As with CPIO, you can give pax a list of filenames to be archived. This list can be created any way you want, including with the `find` command. For example, to generate the list of files owned by bob, modified during the past seven days, and with a character count of less than 500,000:

```
find <startingPlace> -user bob -mtime -7 -size -500000c -print
```

There is plenty of rope to hang yourself with all of the pax options, but it gives you tremendous flexibility. Be careful.

Organizations with large backup needs may want to look at solutions such as AMANDA, the Advanced Maryland Automatic Network Disk Archiver. It is a backup system that enables the administrator of a LAN to set up a single master backup server to back up multiple hosts to a single large-capacity tape drive. AMANDA uses native `dump` and/or GNU `tar` facilities and can back up a large number of workstations running multiple versions of UNIX. See <ftp://ftp.cs.umd.edu/pub/amanda>.

Conclusion

One of the reasons we don't hear of many disk-failure catastrophes is that competent system administrators are quietly doing their jobs. When a disk fails – no big deal – they just go to the backups and get everybody up and running again quickly. The memorable stories are about the few who lose precious information and don't have backup protection. Is that how you want to be remembered?

Software Mini-Review: Red Hat Linux 6.0

by Bob Gray

<bob@boulderlabs.com>

I was excited to receive the new Red Hat 6.0 Linux release. I've been seeking a balanced UNIX package for non-computer experts. This distribution goes a long way toward that goal while still providing full source code for the experts. With the GNOME desktop environment and the Enlightenment Window Manager, a willing novice can quickly become productive and throw Windows out the window. (Actually, there is value in setting up a dual-boot system because of the multitude of Windows applications that don't yet run under Linux.)

Linux distributions have come a long way in a year – especially Red Hat, with the backing of IBM, Netscape, and others. The two printed manuals that come in the box are excellent. The "Getting Started Guide" takes you through the GNOME desktop environment. The "Installation Guide" is well written and easy to follow for the basics. Red Hat defines three classes of installations: Workstation, Server, and Custom. I like their idea of "Workstation-Class" – everything defaults and the installation proceeds automatically. It's a great way for non-Linux people (including myself) to give it a test drive. Of course, there is customization work to be done later, but you have configuration tools to help.

Now here is my *big* gripe – I hope all Source Code UNIX vendors are listening, especially those that hope to expand into the huge market of systems for non-computer people. The installation process for adding Linux to a Win9x PC is miserable. I'm quite familiar with disks, FDISK, partitions, and filesystems, but it still took me a couple of hours of screwing around. In their manual, they talk about this kind of install being the most common, yet they have the audacity to say:

> As a convenience to our customers, we provide the FIPS utility . . . cannot guarantee that FIPS will work properly on your system. Therefore, no installation support whatsoever is available for FIPS; use it at your own risk.

One would think a company recently infused with cash would have rectified this problem by now. What do you expect the "most common installation" user to do? Fortunately, I had a FreeBSD boot disk handy and a second PC. (Their PCMCIA support disk didn't work on a Libretto.) Red Hat could have worked a deal with the PartitionMagic people, developed their own software, or fixed and improved the public stuff. (FIPS is not able to handle a type 14 partition Win95-FAT-16.) So, come on guys; I want to be able to help you get UNIX onto people's Win9x systems. You have got to:

1. document the process completely;
2. provide tools to help the process;
3. support users willing to try it.

(Applause to Caldera for OpenLinux's install process – they have handled the partition splitting problem).

In summary, Red Hat Linux 6.0 is a fine UNIX distribution once the partition problem is handled. If you are reading this article, you probably are already set for your UNIX system and your desktop environment. But take a look at one of the recent Source Code UNIX releases and try the GNOME or KDE. You'll then be in a position to help some less experienced user transition from the mass marketed system to a real operating system.

the trouble with biometrics

by Christopher Calabrese

Chris Calabrese works at an unnamed company at an unnamed job. He has a Master's in computer science from NYU's Courant Institute of Mathematics. His research includes practical applications of security technology.

<chris_calabrese@yahoo.com>

Acknowledgments

The following people provided invaluable help in researching, reviewing, and editing this article: Dario Forte, Ihor Akinshyn, and Marcel Franck-Simon.

In the April 1999 issue of *;login:*, Dario Forte presented the case for authenticating computer users with biometrics – "patterns of unique physiological and behavioral traits"[4]. After I read the article, I began an exchange of email with Dario about some of the little-explored issues with this technology. Our probing took the form of both theorizing and prodding vendors for information. We found some things we really didn't like. This article presents some of them. It also presents some solutions.

The Trouble with Biometrics

Biometrics are nearly infallible at correctly identifying who a person is. But applications don't authenticate people – they authenticate software that sends them some authentication data, like a password. For this to work with biometrics, the application has to ensure that it's talking to a genuine client talking to a genuine biometric device talking to a genuine person. The same problems exist in the password world. In fact, all authentication mechanisms have to deal with the problems of replaying stolen authentication data; stealing authentication data; and trusting the client, the server, and their intermediaries.

So what's the trouble with biometrics? When passwords are stolen, you change your passwords. When credit card numbers are stolen, you have a new card issued. When digital signatures are stolen, you generate new keys. When biometrics are stolen, you can get new ones too. If you're willing to have an organ transplant.

This article examines how well different biometric authentication models prevent biometric data from being stolen and keep from being fooled by stolen biometric data.

The Password Model

The simplest way to perform biometric authentication is to gather biometric data from the end user and ship it off to the application just like a traditional password. To use the real-world example of a bank ATM, the end user would present some body part to the ATM, the client software would send the data to an application across the bank's network, and the application would try to match the biometric data against its user database[8].

This is the most popular model for current biometrically authenticated applications. Unfortunately, it's poorly suited to biometrics. This section examines where biometric data could be injected or stolen along the way.

The Biometric Devices Themselves

The biometrics manufacturers seem to have paid lots of attention to getting high-quality biometric data that comes from a real live person. This is, after all, the area of research from which they sprang. That said, there are two avenues of attack open against these devices. One is device disassembly and modification. This can be dealt with by encasing all the electronics in epoxy, or by using small devices over which the end user has physical control. The other avenue of attack is a Trojan-horse device. Simple device swapping can be dealt with by having the device digitally sign its transmissions with a mechanism like RSA or Digital Signature Standard[10, 11]. This way the device is authenticated to the application along with the user. The device's signature has to be on file along with the user's biometric data, but that's no big deal. I'm not

aware of any device that can do this at this time, however. Also, if the manufacturer or supplier supplies the Trojan then all bets are off – a good reason to stick to trusted suppliers with lots of liability insurance.

Stick to trusted suppliers with lots of liability insurance.

The Biometric Device-to-Computer Link

Most current biometric devices are linked to a control computer by a serial cable or by Ethernet. When I asked vendors about keeping this link secure, one answered that the link should be physically secure[14]. That's fine if it's a short serial cable in a bank under armed guard. It's not a very good solution for a PC in a cubicle or a device attached to the building Ethernet.

Another possibility is using encryption to protect the link. Reasonably simple encryption mechanisms could be used to protect the link from having data stolen, and at least one vendor is taking this approach[14]. However, protecting the link from replay attacks in the password-like model under examination requires digitally signing the data.

The Client Software

This is an area where the biometric-device manufacturers have little control and there are few standards. Given how buggy most software is, this seems like an excellent place to steal biometric data. Insecure operating environments compound the problem by allowing an attacker to drop in a Trojan copy of the software with complete control to inject or steal biometric data. Once again, the only solution under this model is end-to-end encryption and signing.

The Client/Server Network Link

When asked about protecting the client/server network link, manufacturers referred to "industry standards"[14]. I assume they were talking about the protean BioAPI™ standards, which will provide " 'hooks' for secure networking and encryption"[1]. What these hooks are or what kind of encryption is supported is not specified, though my guess is that most applications will use Secure Sockets Layer / Transport Layer Security (SSL/TLS)[3, 6].

If this link is properly encrypted, it may be safe from data injection and stealing. However, the industry standards are not in place yet, so this is still a matter of conjecture.

The Server Software and How It Authenticates

In the password-like model we're examining, the bits that finally end up at the server will be the actual biometric data (albeit possibly encrypted and signed). This data is then compared against a database of known users.

One danger here is that buggy software, an insecure operating environment, or Trojan software will allow an attacker to inject or steal data at this point. As with the client software, the buggy nature of software leads me to believe that at least some implementations will be insecure. Trojaned software can be dealt with somewhat by digitally signing data transmitted by the server.

The real danger at this point, however, is with the authentication database. If the database is kept encrypted with a secure hash that is unique to the application, then stealing the database will only allow replay attacks on this application. If the database is kept in the clear, the data can be stolen and replayed to any application.

A challenge/response protocol with digital signing of the responses addresses many of the vulnerabilities of the encryption-protected password-like model.

Summary

The point of using biometrics is to get away from the shortcomings of the password model, yet the model used by most current biometrically authenticated applications *is* the password model. Not only do we get all the problems of the password model, but we can't even change our authentication codes if they're stolen! Encryption can help, and the industry seems to be moving in the right direction, but nobody's yet talked about the kind of two-way end-to-end encrypting, hashing, and signing that is necessary.

A Challenge/Response Protocol

In the password world, a common solution to the shortcomings of the password model is to use a challenge/response protocol. In this model, the actual password (or biometric data) is never transmitted. Instead, the application sends a challenge that can only be answered correctly if the client knows the correct password (biometric data). The best-known example of this is the PPP Challenge Handshake Authentication Protocol (CHAP), which is used to authenticate dial-up network connections[13]. In the biometrics world, the application would send a random challenge and the biometric device would respond, for example, with a secure hash of the biometric data concatenated with the challenge. By keeping the actual biometric data off the wires and away from any components other than the biometric device and the final authentication system, a challenge/response protocol keeps the biometric data from being stolen at intermediate points. This mechanism does not address replay attacks by a Trojan biometric device, so the biometric device still needs to digitally sign its transmissions to authenticate itself. At least one vendor is working on a challenge/response protocol like this for its product. This section examines how this challenge/response model performs at each step of the way, just as we examined the simple password-like model above.

The Biometric Devices Themselves

The devices still need to be physically secure, and they still need to digitally sign their data to help ensure we don't have a Trojan device.

The Biometric Device-to-Computer Link, the Client Software, and the Client/Server Network Link

Since there is no actual biometric data flowing through these intermediate components, it can't be stolen. Using a unique challenge for each authentication deals with replay attacks.

The Server Software

The server software is still a target in this scenario. Since the application doesn't have to sign its transmissions in this model, however, it's easier to separate the authentication component from the rest of the application. This makes it easier to build a secure authentication component.

The Authentication Database

A challenge/response protocol does nothing to deal with the problem of data being stolen from the authentication database.

Summary

A challenge/response protocol with digital signing of the responses addresses many of the vulnerabilities of the encryption-protected password-like model. It also makes it easier to build a hardened authentication component that's separate from the rest of

the application. It should even be easier to implement than a password-like protocol with the necessary encryption, hashing, and signing.

Authenticating Only to a Crypto Card

In the password world, the rung on the ladder above challenge/response is not using passwords to authenticate to applications at all. Instead, passwords are replaced with cryptographic authentication. One of the problems with such systems is keeping your private keys from being stolen. One of the best solutions is never allowing them to leave a crypto card dedicated to performing all the necessary cryptographic computations. One of the problems with this approach is authenticating the user to the crypto card. This is a perfect application for biometrics: no complex operating environments, no network transmissions of biometric data, no big database of biometrics. Just a card with a couple of chips epoxied to it that you keep in your pocket. Devices of this type are starting to come to market, and it's also an active research area[2, 5, 7]. This section examines potential problems with this approach.

Stealing Biometric Data Stored on the Card

This is a fairly easy problem to solve. Store only a secure hash of the biometric data on the card. To perform the authentication, the card hashes the data from the biometric reader and compares it to the stored value. If the values match, you're in. If the card is stolen and the hashed keys are retrieved, they still can't be used for anything.

Store only a secure hash of the biometric data on the card.

All bets are off if the card is a Trojan, but there are a number of reasons a Trojan card attack would be difficult to pull off. First, any Trojan card has to include the real card's keys or the owner will notice pretty quickly that the card's not working properly. This means a successful Trojan is one where the real card is stolen, modified, and returned, or where the manufacturer/supplier supplies the Trojan. The stolen/modified/returned route is pretty unlikely since physical security of such cards is easy and modifying them is not. The manufacturer/supplier is an issue, so, as I said before, make sure you go with trusted suppliers with lots of liability insurance.

Injecting Stolen Data into the Card

The card can be unlocked by only one person, who usually carries the card on his or her person, so you have to steal the card and the right person's biometric data. If you do this, you still have to deal with the fact that the devices can check that they're dealing with a real live person. Sounds mostly like a nonissue, assuming the manufacturer didn't install a back door.

Stealing the Keys on the Card and other Nonbiometrics Issues

Yes, there are known attacks on card devices, though fewer for a device of this type than for a full smartcard[9, 12]. One nagging issue is a rogue client program that gets your card to sign something it shouldn't have. At least the digital signatures remain intact, however, and the biometric data is not compromised at all. Also, if the card is compromised, you can always dispose of it, get a new one, and generate new keys – just as you get a new credit card if your wallet is stolen. Not pleasant, but much better than an organ transplant.

Summary

This approach addresses all the issues with biometrics being stolen or replayed. It introduces some new issues of digital signature management, but the trade-off is probably worth it for many applications.

Two alternatives to the current password-like biometric authentication model: challenge/response and authenticating to a crypto card.

Why This Is Important

- The personal privacy standpoint: If your biometric identity is stolen, you need surgery to get it back.
- Liability standpoint: If somebody steals biometric data using your application or off computers you control, bad things may happen to your relationship with your insurance company and your customers.
- Third-party standpoint: If your customer's biometric data is stolen, you can no longer use that mechanism to authenticate that customer.
- Biometrics vendors' standpoint: If enough people's biometric data is stolen, nobody will buy devices of the type that have been compromised. There may be liability issues too.

Conclusions

Biometric authentication is a great way of authenticating people to computer applications. Most current implementations, however, leave a lot to be desired. What's scary is that some people assume that because biometrics are nearly infallible for identification, applications that use biometrics for authentication are automatically safe and secure. Even scarier is that others don't even care if biometrics are actually safe, since using biometrics is just so cool. There's also the question of whether authenticating a *person* is appropriate for a particular application (think Swiss banking privacy laws).

This article has identified two alternatives to the current password-like biometric authentication model: challenge/response and authenticating to a crypto card. The challenge/response scheme has issues with the final authentication database/process being subverted, while the crypto-card scheme has issues with the client software being subverted. Which one is safer will depend on the nature of the application. Both are superior to any password-like scheme. They're light-years ahead of current password-like implementations, which feature a distinct lack of attention to proper encryption of the data stream and crypto-authentication of the devices.

Let's hope the industry and its customers realize the current dangers before someone steals a large database of biometric data and puts part of the industry out of business at great expense to their stockholders, their insurance companies, and their customers.

References

[1] <http://www.bioapi.org/arch_des.pdf>.

[2] Cascade Project, "CASCADE, A Smarter Chip for Smart Cards," May 1998. Available from <http://www.dice.ucl.ac.be/crypto/cascade>.

[3] T. Dierks and C. Allen, "RFC 2240: The TLS Protocol Version 1.0," Internet Engineering Task Force (IETF), January 1999. Available from <ftp://ftp.isi.edu/in-notes/rfc2246.txt>.

[4] Dario Forte, "Biometrics: Untruths and the Truth," *;login:*, April 1999.

[5] Dario Forte, private correspondence, April 1999.

[6] A.O. Freier, P. Karlton, and P.C. Kocher, "The SSL Protocol, Version 3.0," Netscape Communications, March 1996. Available from <http://home.netscape.com/eng/ssl3/ssl-toc.html>.

[7] U.S. General Services Administration, "GSA Office of Smart Card Initiatives – Overview and Concepts," May 1998. Available from <http://policyworks.gov/org/main/me/smartgov/initiatives/scpaper.htm>.

[8] Terri Langford, "Texas Bank Uses Eye Scan to Get Cash," Associated Press, May 1999. Available from <http://dailynews.yahoo.com/headlines/ap/technology/story.html?s=v/ap/19990514/t c/atm_eye_scanners_5.html>.

[9] T. Messerges, E. Dabbish, and R. Sloan, "Investigations of Power Analysis Attacks on Smartcards," USENIX Workshop on Smartcard Technology, May 1999.

[10] U.S. National Institute of Standards and Technology, "Announcing the Standard for Digital Signature Standard (DSS)," Federal Information Processing Standards Publication 186, May 1994. Available from <http://www.itl.nist.gov/div897/pubs/fip186.htm>.

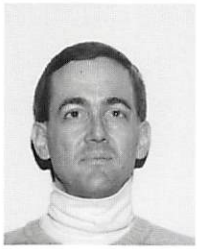
[11] R. L. Rivest, A. Shamir, and L. Adleman, "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems," *Communications of the ACM*, February 1978.

[12] B. Schneier and A. Shostack, "Breaking Up Is Hard to Do: Modeling Security Threats for Smart Cards," USENIX Workshop on Smartcard Technology, May 1999. Available from <http://www.counterpane.com/smart-card-threats.html>.

[13] W. Simpson, "RFC 1994: PPP Challenge Authentication Protocol (PPP)," Internet Engineering Task Force (IETF), August 1996. Available from <ftp://ftp.isi.edu/in-notes/rfc1994.txt>.

[14] Private correspondence with various biometrics vendors, April 1999.

java performance



by Glen McCluskey

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ performance, testing, and technical documentation areas.

<glenm@glenmccl.com>

Since Java first came out a few years ago, debate and discussion about performance have continued, especially in terms of comparison to other languages. Java has the philosophy of "write once run anywhere," that is, the idea of compiled Java programs being portable. This portability is achieved by compiling Java programs into bytecodes, which are then interpreted at runtime. More recently, there has been an emphasis on use of just-in-time compilers, by which bytecodes are compiled at runtime into a more efficient native form.

It is interesting to look at the current state of Java performance. One way is to devise some simple benchmark programs. Such programs can provide useful information and also have some limitations.

Sorting

The first benchmark program to consider is one that sorts a vector of numbers, using an O(N**2) algorithm. This program doesn't have any dependencies like I/O or use of library functions, and is valuable in measuring the actual execution speed of Java bytecodes. A similar program can be written using C++, as a comparison. The Java program looks like this:

```
public class sort {
  public static void main(String args[]) {
    final int N = 25000;
    short vec[] = new short[N];
    for (int i = 0; i < N; i++)
      vec[i] = (short)(N - i);
    for (int i = 0; i < N - 1; i++) {
      for (int j = i + 1; j < N; j++) {
        if (vec[i] > vec[j]) {
          short t = vec[i];
          vec[i] = vec[j];
          vec[j] = t;
        }
      }
    }
  }
}
```

and its C++ counterpart:

```
const int N = 25000;
short vec[N];
int main()
{
  for (int i = 0; i < N; i++)
    vec[i] = N - i;
  for (int i = 0; i < N - 1; i++) {
    for (int j = i + 1; j < N; j++) {
      if (vec[i] > vec[j]) {
        short t = vec[i];
        vec[i] = vec[j];
        vec[j] = t;
      }
    }
  }
  return 0;
}
```

We will use three different compilers for this analysis:

JDK 1.2 (Java 2) interpreter

JDK 1.2 just-in-time compiler

C++Builder 4 (Borland C++)

enabling compiler optimization switches in each case.

The running times of the above program are as follows:

interpreter	326 seconds
just-in-time	8
C++	7

The just-in-time compiler represents a huge advantage over interpretation and is nearly as fast as C++.

But this example also illustrates one of the pitfalls of comparing the performance of two different languages. Suppose that I change the line in the Java sort that reads:

```
for (int i = 0; i < N; i++)
```

to:

```
for (int i = 0; i <= N; i++)
```

This off-by-one error will be caught when the Java program runs, but an equivalent error in the C++ program probably will not be. In other words, Java checks array subscripts at runtime, but C++ does not. The Java approach does more, but at some cost. Choosing which one is "right" depends a lot on your philosophy.

The Java approach does more, but at some cost.

Copying Files

For the second example, consider copying files. The Java version is:

```
import java.io.*;
public class copy {
  public static void main(String args[])
  {
   if (args.length != 2) {
    System.err.println("error");
    System.exit(1);
   }
   try {
    FileInputStream fis =
     new FileInputStream(args[0]);
    FileOutputStream fos =
     new FileOutputStream(args[1]);
    byte buf[] = new byte[1024];
    int n;
    while ((n = fis.read(buf)) > 0)
     fos.write(buf, 0, n);
    fis.close();
    fos.close();
   }
   catch (IOException e) {
    System.err.println(e);
    System.exit(1);
   }
  }
}
```

and the C++ version:

```
#include <stdio.h>
int main(int argc, char* argv[])
{
  FILE *fpin, *fpout;
  if (argc != 3 || (fpin = fopen(argv[1], "rb")) == NULL ||
   (fpout = fopen(argv[2], "wb")) == NULL) {
    fprintf(stderr, "error\n");
    return 1;
  }
  int n;
  char buf[1024];
  while ((n = fread(buf, sizeof(char), sizeof(buf), fpin)) > 0)
   fwrite(buf, sizeof(char), n, fpout);
  fclose(fpin);
  fclose(fpout);
  return 0;
}
```

We could have used system calls (`read`/`write`) in the C++ version, but it seems fairer to use `fread`/`fwrite`, in that the Java I/O library, even at its most fundamental level, is at least one layer removed from actual system calls, and so the C++ version should be as well. Running times for copying large (125 MB) uncached files are:

interpreter	38
just-in-time	40
C++	37

In other words, there's not really much of a difference between the approaches. We might expect this similarity, in that `FileInputStream.read()` and `FileOutputStream.write()` are native methods, meaning that they are implemented in terms of underlying functionality written in some other language, for example, C. In other words, low-level Java I/O immediately translates into calls to underlying libraries, somewhat like `fread`/`fwrite` calls being implemented in terms of `read`/`write` system calls.

Tabulating Word Frequencies

The final example is one that uses higher-level library routines to tabulate word frequencies. Input is a file containing a list of words, one per line, and the output is a list of words and their frequencies. In Java, one way of implementing this is:

```
import java.io.*;
import java.util.*;
public class count {
  public static void main(String args[])
  {
   if (args.length != 1) {
    System.err.println("error");
    System.exit(1);
   }
   HashMap hm = new HashMap();
   class Rec {int cnt;}
   try {
    FileReader fr = new FileReader(args[0]);
    BufferedReader br = new BufferedReader(fr);
    String s = null;
    while ((s = br.readLine()) != null) {
     Rec r = (Rec)hm.get(s);
```

```
      if (r == null) {
       r = new Rec();
       hm.put(s, r);
      }
      r.cnt++;
     }
     br.close();
    }
    catch (Throwable e) {
     System.err.println(e);
     System.exit(1);
    }
    Iterator iter = hm.entrySet().iterator();
    while (iter.hasNext()) {
     Map.Entry e = (Map.Entry)iter.next();
     System.out.println(((Rec)e.getValue()).cnt +
      " " + e.getKey());
    }
   }
  }
```

and in C++, we would say:

```
#include <fstream>
#include <iostream>
#include <map>
#include <string>
using namespace std;
int main(int argc, char* argv[])
{
  if (argc != 2) {
   cerr << "error" << endl;
   return 1;
  }
  ifstream infile(argv[1]);
  if (!infile) {
   cerr << "error" << endl;
   return 1;
  }
  typedef map<string,int> MAP;
  MAP hm;
  string s;
  while (getline(infile, s))
   hm[s]++;
  for (MAP::const_iterator iter = hm.begin();
   iter != hm.end(); iter++)
    cout << iter->second << " " << iter->first << endl;
  return 0;
}
```

In both cases, we use hash tables to accumulate the words and their frequencies. For a list of around 900K words, running times are:

interpreter	86
just-in-time	27
C++	25

When we compare languages with a benchmark such as this one, a more complex benchmark that uses library features, it becomes harder to analyze and draw conclusions. For example, each program uses a hash-table data structure provided in the standard libraries. There are tradeoffs to be made in designing such structures, such as

When we compare languages with a more complex benchmark that uses library features, it becomes harder to analyze and draw conclusions.

Whether Java will ever be as fast or faster than languages like C or C++ is hard to say; it might be better to ask "Faster at what?"

whether you're trying to optimize for speed or for space. If you're optimizing for speed, you might be more aggressive in growing the table size when the table starts to get full.

One way of interpreting the relative times in this case is to look at this application as one that (1) uses low-level I/O, with similar performance across the three compilers, (2) uses high-level I/O (reading lines), which is affected by the issue of interpreter versus just-in-time versus native-code costs, and (3) uses hash-table data structures, with tradeoffs in representation and speed versus space.

Other Performance Areas

There are other performance areas we've not really looked at. For example, an important one in some contexts is invocation time. Using the above compilers, it takes about 15 milliseconds to invoke a null C++ program, and 325 for a null Java one. Another area that might be looked at is GUI performance.

Conclusion

The whole area of Java performance is changing rapidly. For example, Sun recently introduced its HotSpot compiler, and several vendors are working on native-code compilation for Java. Whether Java will ever be as fast or faster than languages like C or C++ is hard to say; it might be better to ask "Faster at what?" or ask whether taking a slight performance hit in return for added functionality (such as array subscript checking) is a worthwhile tradeoff to make. We already make such tradeoffs in other areas today, for example in using high-level languages in preference to assembly language.

using java

Input and Output Streams

Input and output are clearly important parts of any language. In fact it is very likely that one of the first features of any new language that a practitioner learns is how to do IO. An example of this is the well-known "Hello World" example, which relies on being able to pipe the characters to an output "stream." Further, the ability to perform IO is a crucial part of any serious debugging effort. The ubiquitous "printf" is probably the first debugging "tool" that programmers use. In Java, all IO is accomplished using streams. The application creates a stream (which is not unlike a UNIX file descriptor) and then performs operations on the contents of the stream. This article is a short tutorial on "streams" – the basis for IO in Java.

by Prithvi Rao

Prithvi Rao is the founder of Kiwilabs, which specializes in software engineering methodology and Java training. He has worked on the development of the MACH OS and a real-time version of MACH, and he holds two patents resulting from his work on mobile robots.

<prithvi+@kiwilabs.com>

I'll begin by describing input and output streams and then present examples of using them to implement simple UNIX utilities such as `cat` and `ls`.

Input Streams

The class `java.io.InputStream` is the base class for creating input streams. We will look at details of this class.

The sources of some typical input streams are File, String, Array of Bytes, and Sockets.

This is an abstract class and so we must create a subclass (also called a derived class); in other words, we cannot use it "as is." The methods in this abstract class describe the properties of the input stream. Subclasses of `java.io.InputStream` can override some or all of its methods.

Remember that all packages in Java will have exceptions associated with them, and `java.io.InputStream` is no exception (pardon the pun). In fact, almost all of the methods will throw `IOException`. The class definition will generally tell you when the exception will be thrown.

The first form of the input stream stores a byte (having read it from the stream) and places it in a buffer.

```
public int read(byte[] b) throws IOException, NullPointerException
```

The buffer `b` must have been created with `new`; otherwise a null pointer exception results. This method tries to fill the buffer and returns the actual number of bytes read.

The second form of this method reads `len` bytes into the buffer starting at the position specified by `offset`.

```
public int read(byte[] b, int offset, int len) throws IOException,
  NullPointerException, IndexOutOfBoundsException
```

Notice that the last exception is raised if reading takes it out of bounds.

Another useful method that is part of this package is `skip`. It attempts to skip `n` bytes. It may not always be able to do this (for instance, if EOF is reached).

```
public long skip(long n) throws IOException
```

It discards the bytes skipped, where `n` is the number of bytes to be skipped, and it returns the actual number of bytes skipped.

It is also possible to query the input stream for available bytes. The following method returns the number of available bytes from the input stream.

Marking streams is useful for things like parsers that have to look ahead to see what is in the stream.

```
public int available() throws IOException
```

The interesting thing about this method is that the next caller can expect to read that number of bytes returned from `available` without blocking.

There is a way to "remember" all bytes read after marking an input stream.

```
public synchronized void mark(int readlimit) throws IOException
```

If the `markSupported` method returns true, then `mark` remembers the stream at the position when `mark` is called. It then remembers everything that is read from that position in the stream. It is now ready to resupply those same bytes whenever the `reset` method is called.

However, if the application reads more than `readlimit` bytes before the call to `reset` is made, there is no guarantee that it will supply all the bytes again.

Let us consider the cases when `markSupported` returns true and false.

```
public boolean markSupported()
```

Suppose it returns true. Then `reset` might throw an `IOException` if `mark` has never been set or has not been set since the last call to `reset`. It might throw an `IOException` if more than `readlimit` number of bytes have been read since the last call to `mark`. If no `IOException` is thrown, then the stream is reset to again supply (to any thread) all bytes read since the last call to `mark`.

Suppose it returns false. Then a call to `reset` might throw an `IOException`. If no `IOException` is thrown, then the stream is reset to a fixed state that depends on the type of input stream and how it was created. The bytes supplied after this to any caller of `read` depends on the type of input stream.

In general, marking streams is useful for things like parsers that have to look ahead to see what is in the stream. If the parser doesn't like what it gets, it might throw an exception and try something else with the bytes being supplied again.

Output Streams

The base class for output streams is `java.io.OutputStream`.

The methods of this class virtually mirror those of the input stream. This abstraction makes it easier to remember the semantics of IO in Java.

```
public abstract void write(int b) throws IOException
```

This writes one byte to an output stream. The byte written is the low-order bits of `b`. The 24 high-order bits are ignored. It can throw an `IOException` if the stream is closed.

The second way of writing to the output stream is the following:

```
public void write(byte[] b, int off, int len) throws IOException,
  NullPointerException, IndexOutOfBoundsException
```

This method writes `len` bytes from `b` starting from `off`. It throws an `IOException` if `len + off` exceeds length of the buffer. This method really loops writing a single byte at a time by calling `write(int b)`. Since an `IOException` might be thrown during a write, there is no guarantee that all of the buffer will be written.

The output stream can be "flushed" using:

```
public void flush() throws IOException
```

Examples using `java.io.FileInputStream`

This program creates a command that is similar to the UNIX command `cat`. The `cat` command takes a filename as argument and displays the contents of that file on the terminal.

```
import java.io.*;
```

Import the `io` package, because that's where all the `io` classes are defined.

```
public static void main(String[] args) throws IOException
```

As we have seen, many of the `io` methods throw `IOException`. We don't want to clutter the program with code for exception handling. Instead, we let `main` throw an `IOException`.

```
InputStream ins = System.in;
int abyte;
```

This is the variable for the input stream we will use. We initialize it to be the standard input. If no argument is supplied, this program will read `System.in` and then just echo whatever it reads.

```
if (args.length == 1)
  {
   try
    {
     ins = new FileInputStream(args[0]);
    }
   catch(FileNotFoundException ex)
    {
     System.out.println("Cat: " + ex);
     System.exit(1);
    }
   }
```

If an argument has been provided on the command line, interpret it as a filename and try to open an input stream with that file as the source for the stream. If file is not found, we catch that exception, because that's what the UNIX `cat` command does too.

```
while ((abyte = ins.read()) != -1)
  System.out.print((char)abyte);
System.out.println();
```

The hard work is done and we've opened the stream. Now we just sit inside a loop and read one byte at a time, printing those bytes out as we go.

```
if (ins != System.in)
  ins.close();
```

Close the stream when we are finished.

Example using `java.io.File`

This example demonstrates the use of `java.io.File` to build a very poor version of the UNIX command `ls`, which lists files in a directory.

This command will print the files in the current directory if no argument is supplied. If arguments are supplied, it treats those arguments as files or directories and lists them. This is part of the main program.

```
if (args.length == 0)
  {
   filesOrDirs = new String[1];
   filesOrDirs[0] = new String(System.getProperty("user.dir"));
  }
```

If file is not found, we catch that exception, because that's what the UNIX `cat` command does too.

We use a `public static` method so it could be called by anyone who imported this package.

If no argument is supplied, then the directory to list is the current directory. `getProperty` of the class `java.lang.System` contains many properties, one of which is the current directory.

```
else
  {
   filesOrDirs = new String[args.length];
   System.arraycopy(args, 0, filesOrDirs, 0, args.length);
  }
```

Arguments have been supplied; copy them into an array for later use.

```
for (int i = 0; i < filesOrDirs.length; i++)
  {
   ls(filesOrDirs[i]);
  }
```

Go into a loop that calls the method `ls`. `ls` does the hard work.

We now have to write the `ls` method.

```
public static void ls(String fname)
```

We use a `public static` method so it could be called by anyone who imported this package (if we had put it in a package . . .). Keep in mind that `fname` is a string that is either the name of the current directory or a path.

```
File f = new File(fname);
```

Create an instance of `File` representing the name `fname`.

```
if (!f.exists())
  {
   System.out.println("Ls: " + fname + ": No such file or directory");
   return;
  }
```

`exists` is a method of `java.io.File`. Check to see if this file exists. If not, print message and return. We don't exit the program because that's not how the UNIX `ls` behaves. The UNIX version of `ls` continues even if some files don't exist.

```
if (f.isFile())
  {
   System.out.println(fname);
  }
```

If this is an actual file, it prints the name.

```
if (f.isDirectory())
  {
   String[] flist = f.list();
   File tmp;
```

If it's a directory, then the method `list` returns an array of strings containing the names of all files in the directory. So we now know that `fname` is a directory and we know all the files in that directory.

```
for (int i = 0; i < flist.length; i++)
  {
   tmp = new File(fname, flist[i]);
   if (!tmp.exists())
     {
      System.out.println("Ls: " + flist[i] +
             ": No such file or directory");
      continue;
     }
```

```
    System.out.println(flist[i]);
  }
```

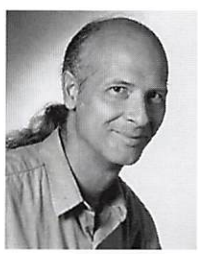
Sit in a loop, create a new `File` object, check that the file exists, and print out the name.

Conclusion

Clearly, the ability to do IO is very important in Java. We have shown how to manipulate a very small part of the overall capability. The beauty of using streams is that once the application has opened a stream to either read or write, the semantics are the same regardless of what is being read from or written to. This makes writing network applications, for example, much simpler than one might suppose.

In future articles we will explore this package further. An interesting exercise for readers is to implement other UNIX-type capabilities to enhance their understanding of this powerful package.

musings



by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V.*

<rik@spirit.com>

I just looked at my calendar. It seemed that I traveled every week since the beginning of April and continued to do so until the third week in June. But I was wrong. I did get to stay home three weeks out of that time. I am really glad that I don't have to travel as much as some people I have met.

During my travels, I had a chance to visit Raleigh, North Carolina. I am sure it is a wonderful place if you live there, but it seemed a bit, well, slow, after spending several days in New York City. The City of Raleigh was the cohost, along with Red Hat, for Linux Expo. Red Hat is the most successful distributor of Linux so far. Other sponsors included S.u.S.E, a German distributor of Linux; and IBM, a surprising contributor to Open Source, as well as being local to the area.

But I didn't come to see Raleigh, or even to meet Bob Young, CEO of Red Hat. I came to see what it was like to be at a Linux conference. Talk about time warps! The level of enthusiasm was reminiscent of the "early" days of UNIX in the '80s. It seemed that the average attendee had less than two years' exposure to Linux and was very excited about anything to do with Linux.

Extreme Linux

A good example of the enthusiasm, and one that gives me shivers of déjà vu, is "extreme Linux." I remember seeing the cover of one of the first *UNIXWorld* magazines. The cover featured a picture of an enormous control console for an early multiprocessor design using a version of UNIX as the operating system. It was entitled "UNIX on Big Iron," although "big iron" is really an affectionate nickname for mainframes (which as of MVS 10 include support for AIX-flavored UNIX).

Extreme Linux covers the use of Linux as the operating system for clusters of computers. All of Saturday was devoted to talks about extreme Linux, which I am not criticizing; rather, I'm commenting on how things have come full circle. Certainly today's

Linux has so far managed to remain a single, mostly coherent distribution. But there are starting to be some exceptions.

technologies, chiefly cheap hardware in the form of PCs, are different from what was available in 1985, but people still get excited about having an enormous amount of processing power coordinated using their favorite operating system. There were more "papers" about clustering (extreme Linux) than any other topic.

I sat in on some of these talks, and wondered (to myself) what it was about Linux that particularly supported the use of clustering. Actually, the only thing I could find by listening or reading the papers was that Linux was chosen on the basis of a groundswell of interest. In one case (the Hebrew University's MOSIX project), Linux was the third OS platform supported, not the first. So, in reality, *BSD and other versions of UNIX were not left out; they just were not part of the conference's focus.

Graphics did come in a close second. Another of the driving forces behind Linux today is the growing popularity of Linux for gaming, and having hot device drivers is critical to the success of Linux as a platform for Quake. For example, Quake can bypass the usual operating-system controls and access memory for graphics cards directly, providing more performance. And, incidentally, a couple of root exploits as well.

I am not criticizing Linux or the Linux community, but rather looking at it as newer, and perhaps young. Enthusiasm is wonderful, so long as we don't go making the same mistakes we did in the past.

Mistakes?

Linux has so far managed to remain a single, mostly coherent distribution. That is, you can expect to be able to compile software written for Linux on any up-to-date distribution, and run any binary compiled for Linux on x86 processors. But there are starting to be some exceptions.

Ted Ts'o, recently at MIT and now at VA Research, gave a nice talk about device drivers and how important it is to standardize. One Linux distribution, Slackware, has shipped old kernel include files, making driver builds fail. Another driver issue that Ted mentioned is that the kernel currently ships with all drivers compiled into the kernel. There is no way to load just the drivers needed, or additional, oddball drivers. This leads to kernel bloat (which is not a problem just for Linux) and the inclusion of possibly buggy drivers into every kernel.

And then there are the desktop wars. As in the UNIX days of yore, there are several popular desktops (window managers) as well as GNOME, a library that supports graphical applications including window managers. GNOME does include other cool features, such as the ability to control resources of active applications and interprocess communication using CORBA (as a stand-in for OLE 2, or Microsoft COM, as it is better known).

Currently, only Enlightenment (or sometimes just E) and one other window manager sit on top of GNOME. On the other hand, there is also KDE, which is something like CDE (the Common Desktop Environment, embraced by many commercial UNIX vendors today) and window managers supported by KDE. If you have loaded Red Hat 6.0, you will get GNOME and E by default. This can cause problems if you then decide to run a KDE-based application, in that the two window subsystems do not communicate resource and other needs well to each other.

At the moment, there are people who are passionate about the use of one or another of these subsystems. But Linux has not become Balkanized as a result – so far. And breaking Linux into fractious groups is certainly a goal for some.

The Opposition

Microsoft has not been resting quietly. Its stock price has fallen almost 20 percent in value, and this is not something that people who have vested interest in the stock will tolerate. Among the offensive tactics, Microsoft funded a "comparative review" of NT versus Linux as Web and SMB servers, with NT outperforming Linux by more than a factor of two. If you don't believe this, well, you shouldn't. The test was performed by Mindcraft, a group that has done this several times so far and gets better assistance tuning NT servers from Microsoft than does the average billion-dollar corporate customer. On the other hand, Mindcraft attempted to use Red Hat's free installation-support line to fine-tune Linux. Jeremy Allison has a nice article that points out some of the errors made by Mindcraft personnel and also includes a few useful hints for tuning NT servers: <http://www.linuxworld.com/linuxworld/lw-1999-04/lw-04-mindcraft.html>.

Microsoft funded a "comparative review" of NT versus Linux as Web and SMB servers, with NT outperforming Linux by more than a factor of two. If you don't believe this, well, you shouldn't.

The infamous Halloween papers recommend that Microsoft "decommoditize" protocols in order to make the problems of competing (as well as cheaper and better performing) OSes go away. What this amounts to is to take standards, like Kerberos V or HTML, and extend them in incompatible ways by bending the standards. Then, keep the changes secret, as well as continue to modify them often enough to prevent imitation. IBM tried these techniques years ago, not long before its stock price tanked, and it looked like IBM might even go out of business. Will history repeat itself? That is, will Microsoft embrace open standards (as IBM apparently has) and recover? I for one am not holding my breath.

The Microsoft antitrust case has resumed as I write this, and I expect history to be repeated in a different sense. That is, the Justice Department will generally leave Microsoft to continue to experiment with its monopoly, which would be bad for the industry as a whole. Why? It will simply take longer to move to open standards, get away from bloatware, and be able to embrace advances in computing from the tens of thousands of programmers involved in open source (as opposed to the hundreds involved in writing NT or Office). Those companies hoping that Microsoft will open up its directory services (the part of Windows 2000 that replaces domain controllers) had better increase their pressure, and soon, if they are to have any hope at all of succeeding.

More Déjà Vu

The Linux Expo exhibit floor was not only large, but also well attended. And when you walked in the door, there were IBM, SGI, DEC/Compaq, and a little Sun booth tucked away toward the back. To me, it was reminiscent of UNIX Expo in New York City. Not as large and without the circus-like atmosphere as vendors put on sideshows (jugglers, actors, singers, and scantily clad young men and women) to attract attention. But still, to see the major UNIX OS vendors at a Linux Expo show was, to me, surprising.

There was still the small-conference atmosphere to some extent. One vendor's booth, Watchguard, was serving good beer and offering free Internet hookups. And, of course, there were the usual free handouts. A nice map would have helped.

I did pick up a copy of Red Hat 6.0 while I was there. I wanted to share it with a local friend who had lent me an earlier version when I discovered that BSD was not going to load on my new notebook.

I have been trying out various OSes on PC hardware, looking for strengths and weaknesses. I got a copy of Solaris 7 for free and loaded it with little trouble. The load was not trouble-free – Solaris had to be in the first disk partition or it wouldn't boot. Sulky software.

A free compiler isn't free when you can't install it.

I was pleased that it recognized most of my hardware, including my monitor's vendor and type. No futzing around with timings and worrying if the settings would destroy my monitor – Solaris just worked. But then, there is no C compiler or Perl. There is a site where you can get free downloads of both; the format used proved unusable, though.

Solaris 7 handles disk partitioning, so that you have lots of partitions. Then the Sun install program wants more disk space in those partitions and will fail with useless error messages when it runs out of space. I tried being creative by using symbolic links to add the illusion of more space whenever a console error message would give me a hint, but I gave up after two hours. A free compiler isn't free when you can't install it.

I needed NT for some testing, so I installed it on the same hardware. It recognized the display adapter, but not the monitor, and failed to recognize the network adapter as well. Installing the network adapter is not hard but just a nuisance. Also, you must manually install the latest Service Pack, even though it really turns NT 4 into NT 4.4 or 4.5. It took more than twice as long to get an installation going, and with only 640x480 resolution.

With the Red Hat 6 package in hand, I tried installing it. Installation was comparable to *BSD versions I have tried, although I made the mistake of trying to use the Disk Druid (best to stick with `fdisk`). But 15 minutes later, Red Hat was up and running, without my having cracked any documentation. Again, `xf86configure` did not recognize my monitor as Solaris did, and it took some tweaking to get X working properly. Then I got to try out E and GNOME, which was different enough from the other window managers that I liked it (especially being able to scoot the bottom panel out of the way).

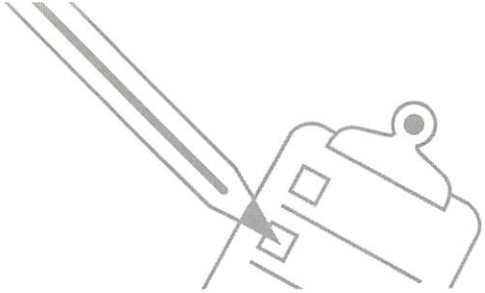
My friend Jay was not so lucky. He let the install script scan for bad blocks, and when the scanning program crapped out, the installation script hung. He also had other disk-related problems because he had a new, large (8.6 GB) hard drive, and the Red Hat install program had some problems with it as well. Finally, booting failed because the SMP version of the Linux kernel boots by default but had trouble finding the second processor on his Tyan motherboard (there wasn't one). Jay did have to read the Red Hat docs and discovered that he could choose to boot the single-processor version.

I have often toyed with the idea of benchmarking NT versus various *nices on the same PC hardware, but am glad that I haven't. Successful benchmarking requires tweaking the operating system perfectly in order to get the best performance. And, generally, benchmark testers may be more or less successful at doing this, skewing the results (the way Mindcraft did).

Jeremy Allison, in the article mentioned earlier, states that the only relevant benchmark is one where you use your own applications in your own environment. Even this benchmark will be prone to some problems, unless you have expert assistance in tuning both the operating system and the applications' behavior.

Still, if you have done this in your organization, I think we would all be interested in the results, as well as in learning what you did to tune the OS and apps. Let me know if you want to write about benchmarks. And don't forget the FREENIX special edition of *;login:* coming early this fall.

standards reports



edited by Nicholas M. Stoughton

USENIX Standards Liaison

<nick@usenix.org>

Those of you who have been following this column over recent months will know about the work of the "Austin Group," a collaboration among the IEEE Portable Application Standards Committee (PASC), The Open Group (TOG), and ISO JTC1/SC22/WG15 to produce a revision of the two core POSIX standards: POSIX.1 and POSIX.2. The result will be a single standard, in four volumes, replacing the old POSIX.1, POSIX.2, and the majority of the Single UNIX Specification (SUS) from TOG.

On June 1, two important events occurred. The first was that the relevant authorities in the IEEE Standards Department and at TOG were able to sign a Joint "Memorandum of Understanding" on the development and publication of this new work. ISO will also become a signatory to this understanding at a future date. However, since the majority of this document deals with intellectual property, and all the intellectual property rights in the base documents belongs to either the IEEE or TOG at present, the lack of an ISO signature should not be taken as an ominous sign, at least not yet!

The second June 1 event was the publication of draft 1 of the new document. At present, only three of the four volumes have been started: Base Definitions (known as XBD), System Interfaces and Headers (XSH), and Commands and Utilities (XCU). The fourth, Rationale (XRAT), is trailing behind since this is a guide, containing no normative requirements, and requires a far higher degree of effort to produce than the base documents. XRAT notwithstanding, draft 1 occupies exactly 2,500 pages. And this is before we have added any of the new material! Reviewing this will occupy much of the time before the next face-to-face meeting of the group, in July in Montreal.

Draft 1 concentrates on taking the existing SUS and presenting it in a new language, with different options from its previous edition. Any interfaces that were marked as legacy or obsolete in the previous editions of SUS or POSIX have been removed. Interfaces that now have better, more consistent, alternatives become "new obsolete"; application developers are being steered away from using these interfaces in favor of the better alternative. Of course, removing an interface from the standard does not remove it from a vendor's implementation; it merely removes the requirement for that vendor to support that interface.

The second draft, due out later this year, should have considerably more functionality from other amendments that are expected to finish before then (e.g., POSIX.1g, Protocol Independent Interfaces, or sockets).

The new draft looks a lot more like the SUS than POSIX.1 or POSIX.2. This is really just a style issue. Rather than grouping interfaces by functional area, as in POSIX at present, all the interfaces are described in alphabetical order. It is felt that this makes it more useful for the application developer rather than the system implementor. Of course, the cynical will feel that this is just TOG taking over and replacing POSIX with its own material, but I would beg patience. Several things have been changed to make the language more precise (e.g., the use of ISO terms such as "shall" rather than the old TOG style of "will") while trying to retain some of the clarity and brevity used in the SUS. For example, there is

The following Reports are published in this column:

POSIX.1h SRASS and POSIX.1m

Whither the X Window Systems?

Our Standards Report Editor, **Nick Stoughton**, welcomes dialogue between this column and you, the readers. Please send your comments to <nick@usenix.org>.

considerable use of shading to show optional behavior, which most people believe shows the intent far more clearly than trying to spell it out in English terms.

While many new options are introduced at draft 1 (several of which may be removed in later drafts), many of these are "mandatory options." This seeming contradiction allows for future profiles of this standard to specify subset behavior (e.g., an embedded system may not require a filesystem), while not allowing a system that claims conformance to the base standard to opt out of critical sections. (Sorry, Microsoft, if you want to claim conformance, you'll just have to implement it in full!)

One current point of contention is in the inclusion of C Standard interfaces. The intent is to have a single point of reference for a programmer to all standardized interfaces required up to the POSIX level. Don't force that programmer to have to go and look elsewhere for details on a required interface, and present all interfaces in the same style. Several of the interfaces defined by the C Standard are extended by POSIX, and so the inclusion of these is easy to justify. But should we include the entire definition of, say, `fopen`, or just the extensions to it? The current stand taken is to reproduce the entire interface, but with a warning that if the description conflicts with the C Standard, then this was unintentional, and the C Standard wins. However, this will probably be an item that goes to formal vote of the working group, since there are certainly individuals who believe that this is the wrong way to go about it.

Alongside the Austin Group is a second informal body, known as the Joint Procedures Committee (JPC). The JPC has built a set of rules by which the working group should conduct its business. The basic principle is to achieve consensus around the table. Each of the three organizations taking part (IEEE, TOG, and ISO) has a nominated Organizational Representative (OR). Since the entire principle is to produce a standard that all three organizations can adopt, it is hoped that there will be few points of serious contention. Any individual who believes that a particular problem exists should discuss this with the group as a whole. If necessary, the individual can approach his or her OR to call for a formal vote. Votes are then taken by Organization, with exactly three votes cast. Abstentions are not allowed. In almost all cases, the votes will have a 45-day period, so that each organization can conduct a letter ballot of its members (usually by email).

This is a very exciting time in the Open Systems Standards world. You are strongly encouraged to visit <www.opengroup.org/austin> and join the discussions and reviews.

POSIX.1h SRASS and POSIX.1m Checkpoint Restart

Helmut Roth <hroth@ nswc.navy.mil> *reports on the April 1999 meeting in Charlotte, NC.*

The POSIX.1h Services for Reliable, Available and Serviceable Systems, (SRASS) and POSIX1.m Checkpoint Restart working groups are in the process of developing a set of APIs for fault management and serviceability applications. The goal of the SRASS Working Group is to support fault-tolerant, serviceable, reliable, and available systems in a useful, portable way. Where feasible, POSIX.1h also needs to be useful for general applications such as distributed parallel database transaction systems and safety-related systems. The work of Checkpoint Restart overlaps this goal, and so the working groups have effectively merged while still producing two separate standards.

The most recent Checkpoint Restart ballot achieved 59 percent approval, and the Working Group is currently resolving ballot objections and revising the draft. It is hoped that the next draft will be out for recirculation soon.

The SRASS document contains several groups of interfaces:

- Logging interfaces, aimed at allowing an application to log application-specific and system events to a system log and subsequently to process those events. Fault-management applications can use these interfaces to register for the notification of events that enter the system log, providing a rudimentary event-management system. Notifications provide a way to proactively manage problems and initiate steps to prevent a system failure later. We now have copyright release from The Open Group to include `syslog()` to support backward compatibility. (Thanks to all of you who helped on this matter.)
- A core-dump control interface to enable an application to specify the pathname to a core-dump file to be used if the process terminates abnormally.
- A shutdown/reboot interface supporting several options such as fast shutdown, graceful shutdown, rebooting with optional scripts, and so on.
- Configuration-space management interfaces, intended to provide a portable method of traversing the configuration space and for manipulating the data content of nodes in that configuration space. This interface will provide a fault-management application access to underlying system-configuration information and the means to direct reconfiguration of the system. This has been changed from a tree-traversal mechanism to a directed graph, since this better represents the complex interrelations among configuration items.

The working group has yet to finalize the changes in the SRASS draft (draft 4.0).

Additional meetings were planned for early June to finish ballot resolution on the previous draft and to get draft 4.0 completed.

If you are interested in helping support fault management (including serviceability and fault-tolerance aspects of systems), please get in touch with Helmut Roth <hroth@nswc.navy.mil> or Dr. Arun Chandra <achandra@vnet.ibm.com>. A mailing list for this work also exists at <srass@pasc.org>; to subscribe send email to <srass-request @pasc.org> with the body "subscribe".

Whither the X Window System?

Finnbarr P. Murphy <fpm@zk3.dec.com> *writes on recent developments in the organization behind the X Window system.*

The X Window System (also known as X11 or, more simply, X) is the de facto standard graphical environment on UNIX and Linux platforms. It has been ported to over 100 other computing platforms. It was originally developed by the Athena project at the Massachusetts Institute of Technology and was first commercially released in 1986. In 1988, the X Consortium was formed to further the development of the technology in an independent vendor-neutral consortium. In the years that followed, many successful versions of the X Window System were released. Since its inception, the X Window System has become one of the most successful consortium-developed, open standards–based technologies ever.

Stewardship of the X Window System technology has passed through a number of hands in the last few years. Upon the demise of the X Consortium in 1986, the technology was handed off to the Open Software Foundation (OSF) which, in turn, was absorbed in 1987 into The Open Group (TOG). The Open Group continued the maintenance and enhancement of the X Window system via two working groups: the X Project Team and the X Standards Team. However, in mid-1998, The Open Group terminated the X Window System project when it decided to close down the OSF facility in Cambridge, Massachusetts.

With the closing of OSF, many companies that were members of the X Project Team came together to form X.org, an autonomous organization under the auspices of The Open Group. Founder (and executive-level) members include Compaq, Sun, Hewlett-Packard, IBM, and SGI. The mission of X.org is the collaborative stewardship and promotion of the X Window System technology, collaboration on bug fixing and code maintenance, maintaining the sample implementation and related standards, and the engineering of appropriate enhancements driven by current and future market requirements.

Three levels of membership are available. Executive membership has the most privileges, including all releases and patches tested on a reference platform, direct access to the engineering contractor for bug fixes, a seat on the executive board, hotline support from the contractor, and much more. Premier membership is the middle tier, at which most non-OSV companies participate. Premier members include Attachmate, Barco, Jupiter Systems, Metro Link, Mitre Corporation, Sequent, Siemens, WRQ, Starnet, and Xi Graphics. Associate membership, entry level, provides companies and individuals with limited budgets a voice in the affairs of X.org.

X.org will periodically provide official X Window System releases to the general public free of charge. One annual major release each February is planned, with three patch releases (one per quarter) between each major release. As part of their membership benefits, X.org members will get the new release ahead of the general public. How far ahead of the general public members will receive this release depends on the level of membership held.

The new Web site for X.org is <http://www.x.org>. A recruitment drive for additional members is under way. X.org held its latest meeting at the 5th Linux Expo in Raleigh, North Carolina, and signed up a number of influential companies. Following an open request for proposals to provide software-engineering services, Metro Link of Fort Lauderdale, Florida, was selected as the contractor for fixing bugs, maintaining and enhancing the sample implementation, maintaining the reference platforms, and preparing the annual release and quarterly patch releases.

Several task forces have already been established:

- Accessibility
  This task force is charged with developing an extension to the X Window System to provide better support for people who are visually impaired or who have other physical challenges that require nonstandard user input mechanisms.
- Audio
  This task force is charged with examining how best to support audio within the X Window System.
- Keyboard and Mouse
  This task force is looking at ways to provide better and more consistent support for keyboards, mice, and other input devices (including mice with scroll-wheels and USB devices).
- Standards
  This task force governs the evolution of the X11R6 specification, working with appropriate groups to revise and post updates to the specifications as required. A revision of the X11R6 specifications is in progress to include the new features provided in X11R6.4 and to remove defunct, obsolescent, or unsupportable programming interfaces. Where appropriate, the output of the other task forces will become input for the standards track.

- Desktop
  This task force is charged with examining the various desktops such as CDE, KDE, and GNOME and figuring out whether the X Window System needs to provide better support for these desktops and, if so, how that support should be provided.
- Fonts
  This task force is charged with examining new font technologies for the X Window System (True Type fonts, Open Fonts, anti-aliased fonts, etc.) and engineering appropriate support for selected font technologies into the sample implementation.

A number of challenges face the new organization. Of primary importance is repairing the damage caused by the decision of The Open Group to introduce a commercial license for the X11R6.4 release in early 1998. The X Window System was one of the first software packages to have what is nowadays known as an open-source license. This liberal license facilitates the existence of many free or low-cost implementations of the X Window System and has contributed enormously to the growth in popularity of the X Window System and, more recently, the Linux operating system.

The introduction of a commercial license immediately caused a splintering of the X Window System development community. The Xfree86 Project <http://www.xfree86.org> is a nonprofit organization that manages the development of a version of the X Window System called XFree86, which runs on a wide range of Intel IA32 (and some other) platforms and various operating systems, including OS/2. This organization traditionally took the latest release of the X Window System sample implementation and rapidly merged it into their existing code base. The Xfree86 project decided to remain with the unencumbered X11R6.3 code. Recently, talks took place between the two organizations and it was agreed that the two would align their code bases and work together in the future to further the X Window System technology.

With the emergence and exponential growth of Linux and other open-source operating systems, the X Window System technology is today enjoying a major renaissance, and a number of new window managers, toolkits, and desktop managers have recently been released.

Following the late-1980 GUI wars between Motif and OpenLook, Motif emerged as the de facto standard window manager on UNIX platforms. However, Motif is not open source; it is a commercial product licensed and sold by The Open Group. The unit royalty fee charged is such that it is uneconomical for Linux operating-system vendors such as Red Hat or Metro Link to distribute Motif in their base product. As a result, the open-source community has been active in developing a number of alternative (and often incompatible) window managers and toolkits including LessTif, Qt, AfterStep, BlackBox, Enlightenment, FVWM, KDE/KWM, SCWM, QVWM, and Window Maker.

This is not necessarily good for application portability. Motif is still the de facto standard window manager for the X Window System. For this reason, X.Org is actively working with The Open Group to develop a license mechanism to deliver an open-source version of Motif for open-source platforms, which X.Org will then manage. The hope is that if a Motif open-source license is available, application developers will choose to remain with Motif.

X.Org has decided that it is not in the business of specifying the desktop metaphor. Rather, it wishes to provide the stewardship of the "plumbing" behind the desktop, that is, the X Window System and Motif technologies. X.Org feels that it is in the best interests of users if innovation in the desktop space is allowed to continue unfettered. GNOME <http://www.gnome.org> and KDE <http://www.kde.org> are examples of excellent innovation in the desktop space that provide users with far better functionality than is available in Common Desktop Environment (CDE).

the bookworm

by Peter H. Salus

Peter H. Salus is a member of ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He has held no regular job in the past lustrum. He owns neither a dog nor a cat.

<peter@pedant.com>

Light Reading

At a time of year when most folks are perusing mysteries or science fiction, here's some less-than-typical summer reading. (By the way, I trust that most of you are aware of the fact that Neal Stephenson, Bruce Sterling, and Vernor Vinge all have new novels out. Stephenson's *Cryptonomicon* is nearly a thousand pages, but quite fine; Sterling's *Distraction* is far shorter and portrays a dystopic future; Vinge's *A Deepness in the Sky* is a "mere" 600, and well worth the reading, continuing the universe that we met in *A Fire Upon the Deep*. If you don't want to read computer stuff at the beach, try these.)

If you want something a bit on the light side, try *Open Sources*. In 14 chapters by (alphabetically) Brian Behlendorf (Apache) to Bob Young (Red Hat), by way of McKusick, Stallman, Torvalds, Vixie, and Wall – and two contributions by Eric Raymond – the field is limned quite well. Despite some strange historical howlers in several essays, this is well worth the couple of hours you'll put into reading it. The GPL is in an appendix; I wish several of the other "free" licenses were there too. A tip of my sun visor to Bruce Perens for noting, "The concept of free software is an old one." (In fact, it precedes the FSF by just under 30 years – SHARE was founded in 1955.)

Staying Mobile

When I first saw a computer, over 40 years ago, it was an enormous monstrosity. Today every other person on an airplane is working on a notebook, and PalmPilots are as ubiquitous as their price permits. Mobility is here. For example, Charles Perkins was on the program committee for the first USENIX Symposium on Mobile and Location-Independent Computing (August 1993); I reviewed his *Mobile IP* about a year ago. His paper on "Mobile Networking with Mobile IP" sits nearly in the middle of this bulky, annoying anthology. My guess is that ACM Press and Addison-Wesley get the blame for this. The articles have been simply transferred to the anthology. Thus, some are single-column, some are double-column, a few are triple-column. There is a rich variety of type styles. The net result is disruptive to any serious reader. But I'm certain that production was cheap. With this sort of formatting, the most gripping work can become unreadable. The title pages are inconsistent, too. The content is first rate, the look-and-feel is just dreadful. The field, the editors, and the authors deserve better for $40.

More Networking

Most of the books I look at just weigh too much. I really don't believe that most topics require 600-1000 pages. [*Sic*! See below – Ed.] And even more aggravating are the myriad volumes with wide margins, lots of boxes and screen shots, and about 100 pages of text.

So when I received Stewart's *BGP4*, I was delighted: under 200 pages and devoted to a single topic! (BGP is the Border Gateway Protocol [RFC 1654, July 1994] derived from RFC 1267, October 1991. BGP is the descendant of EGP [Exterior Gateway Protocol, RFC 904, July 1994; which grew out of RFCs 827 and 888].) *BGP4* is both an easy read and a solid piece of work.

Before I was done with Stewart, two more small books in the series had arrived: Geoff Mulligan's *Removing the Spam* and Andrew F. Ward's *Connecting to the Internet*. Neither of these is up to

Books reviewed in this column:

Chris di Bona et al., eds.
Open Sources
Sebastopol, CA: O'Reilly & Associates, 1999. Pp. 272. ISBN 1-56592-582-3

Dejan Milojicic et al., eds.
Mobility
Reading, MA: Addison-Wesley [ACM Press], 1999. Pp. 682. ISBN 0-201-37928-7

John W. Stewart, III
BGP4
Reading, MA: Addison-Wesley, 1999. Pp. 160. ISBN 0-201-37951-1

Geoff Mulligan
Removing the Spam
Reading, MA: Addison-Wesley, 1999. Pp. 240. ISBN 0-201-37957-0

Andrew F. Ward
Connecting to the Internet
Reading, MA: Addison-Wesley, 1999. Pp. 190. ISBN 1-201-37957-0

Brian Tung
Kerberos
Reading, MA: Addison-Wesley, 1999. Pp. 176. ISBN 0-201-37924-4

Marcus Goncalves
Linux at Work
New York: Wiley, 1999. Pp. 363 + CD-ROM. ISBN 0-471-33349-2

Michael Hiltzik
Dealers of Lightning
New York: HarperCollins, 1999. Pp. 425. ISBN 0-88730-891-0

I. Bernard Cohen
Howard Aiken
Cambridge, MA: MIT Press, 1999. Pp. 412. ISBN 0-262-03262-7

I. Bernard Cohen and Gregory W. Welch, eds.
Makin' Numbers
Cambridge, MA: MIT Press, 1999. Pp. 320. ISBN 0-262-03263-5

the standard of Stewart. In fact, I think that I still prefer Schwartz and Garfinkel's *Stopping Spam* – and they cost pretty much the same.

Connecting to the Internet is a good enough book, but it's hard to see the same level of sysadmin or networker wanting it and the Stewart. It's a solid presentation of well-known material.

I then received Brian Tung's *Kerberos*. It is not merely the first book on Kerberos, it's a real keeper. Tung is the author of the online Moron's Guide to Kerberos and has done a splendid job explaining this authentication system. *Kerberos* is detailed and interesting. Tung and Stewart are definite stars to me; Mulligan and Ward will depend on your level and what you want to get from them.

But there's one striking thing: Each of them is under 300 pages, and two of them are under 200! Great stuff! A good series beginning. Perhaps there'll be volumes on Wave Division Multiplexing and on hubs and routers, too.

More Linux

Goncalves has put together a neat small book. It's not clear to me that it'll be used a lot for "building strategic applications for business," but it is a useful introduction, the CD contains Red Hat 5.2, and the Appendices (which occupy nearly 40 percent of the book) are very useful.

More History

Over 25 years ago, there were no PCs of any kind, but then the Alto – arguably the most important result of the Xerox Palo Alto Research Center, was born. And the entire world has been different ever since. Hiltzik has done a wonderful job in bringing PARC and its staff to life and making the developments comprehensible. At times, he may be a bit overenthusiastic and sound gullible, but he's right on top of all the folks and their products.

Long before there was an Alto, almost 40 years before, there was the Harvard Mark I (planned in 1937 and operating in 1944) – the IBM Automatic Sequence Controlled Calculator. The designer, fundraiser, etc., of the Mark I was Howard Aiken. I. Bernard Cohen, who knew Aiken, and is now in his mid-80s, has produced a combination of biography and technical history that is a really compelling read. I recommend it to anyone interested in the early, electromechanical computers. Cohen has also added a number of inaccessible documents in appendices. Among these are the remarks Aiken made at the "Dedication" of the Mark I and his memorandum of 1949 describing the "Harvard Computation Laboratory." Even more documents of Aiken's are accessible in a volume edited by Cohen and Gregory Welch. My gratitude goes to MIT Press for making them available.

book review

Peter H. Salus, series editor-in-chief

Handbook of Programming Languages

Macmillan Technical Publishing, 1998.
ISBN 1-578-70134-1.

Reviewed by Glenn Vanderburg

<glv@vanderburg.org>

The *Handbook of Programming Languages* – which I will refer to as the *Handbook* – is by any measure an ambitious project. Four volumes, about 2,000 pages, 50 languages (if you count the really little ones), 49 authors (many of them the designers of the languages in question), or almost $200 – whatever your metric, it's a big project. Reviewing it is no small task, either. After a quick overview of the set as a whole, this review discusses each of the volumes in turn and concludes with some comments that apply to all of the books.

The volumes are divided roughly along the lines of different kinds of languages:

Volume I: Object-Oriented Programming Languages

Volume II: Imperative Programming Languages

Volume III: Little Languages and Tools

Volume IV: Functional and Logic Programming Languages

I was surprised a few times by where particular languages appear in this lineup, but for the most part the divisions make sense.

I believe a book review should tell readers why they should or should not buy a book. In this case that's an unusually complex and slippery question. When Salus announced the publication of the *Handbook* in *;login:*, he said, "Ask your corporate or University library to buy a copy." I think that most libraries used by

programmers should have a copy of this work, so Salus's recommendation is a good one. However, some individuals may want to purchase one or more of these volumes. Depending on your needs, these books may or may not be worth buying.

Why would you want to have four big books about a bunch of programming languages? I can think of four reasons:

You could be a strange aficionado of programming languages as I am. If so, the *Handbook* is almost inevitably going to end up on your bookshelf.

You might want a quick introduction to languages you may encounter in your work (if, for example, you are occasionally called upon to maintain or enhance – or merely decipher – existing programs). The *Handbook* is adequate for this need, but the curious omission of some widely used languages, most notably COBOL, will be frustrating.

You might want a survey of languages to help you choose the right language for your development projects. If you fit this description, these books are particularly well suited for you, since most of the current languages that are anywhere close to the mainstream are covered in detail. Even though the authors are closely associated with the languages they write about, nearly all of them have kept their advocacy to reasonable levels, so fair evaluations of the languages are possible.

Finally, you might want to learn about different ways of thinking about problems and expressing their solutions, in the hope that this will make you a more creative problem solver or have some similar benefit. For you, the *Handbook* will be a feast that ends too soon, before you get to some of the really mind-stretching languages.

In reading these books, I looked for several kinds of information about each language:

- historical relationships with other languages
- a concise statement of the design philosophy
- a quick summary of distinguishing characteristics
- enough details to permit reading most programs easily
- examples and discussion that give a good taste of the *feel* of programming in the language – coding idioms, styles of problem solving, overall program structure

In most cases I was pleased by what I found. I did not consider exhaustive syntactic or semantic detail to be particularly important; these books make no claim to be complete language references.

Volume I: Object-Oriented Programming Languages

Volume I begins with an essay on object-oriented programming and a paper about the application of design patterns and frameworks. The introductory essay is by Timothy Budd, and it's hard to think of a better choice. In just a few pages, Budd provides a well-written, easy but thorough overview of object-oriented programming. In keeping with his excellent text on the topic, he manages to keep the discussion free of the details of particular languages, mentioning three (Simula, Smalltalk, and C++) solely to provide historical context. Doug Schmidt's paper about patterns is also good, but it focuses too much on a single application domain (network communication software).

Six object-oriented languages are covered in this volume: Smalltalk, C++, Eiffel, Ada95, Modula-3, and Java. Smalltalk, C++, and Java each merit three chapters, the others only one apiece. With the exception of Ada95, the designers of these languages have made substantial contributions to the book.

The Smalltalk and C++ sections are particularly strong, with excellent coverage of not only the languages themselves, but also the histories and user communities. Bjarne Stroustrup has contributed his usual thorough history and self-critique of C++. My biggest complaint about these chapters is that one of Smalltalk's noted strengths, GUI development, is barely mentioned.

Bertrand Meyer's chapter on Eiffel successfully explains Eiffel's design as a part of the overall software lifecycle. The bulk of the chapter, however, spends too much time on detailed exposition of language features and fails to convey the feel of the language.

The biggest disappointment to me in the entire four-volume set is the chapter on Modula-3. Contributed by Fashad Nayeri, the chapter is a lightly edited version of the Modula-3 language specification. Modula-3's designers are quite proud of the brevity of the language definition, citing this as evidence of the language's simplicity. It is indeed a delightful, simple language, but the terse spec makes difficult reading for programmers and provides very little understanding of what it feels like to program in Modula-3. (In fairness, a colleague read the chapter and felt that he learned a lot about what the language is like. It may or may not be relevant that he has a fair amount of Modula-2 experience.) Modula-3 is less well known than it deserves to be, especially given its influence on more popular languages such as Java. It's unfortunate that most readers will find it difficult to learn about Modula-3 from this chapter.

The section on Java and the chapter covering Ada95 are good but not particularly noteworthy. The only real problem is the lack of historical information for Java. Java's designers have been very quick to acknowledge that Java is mostly a collection of good ideas from other languages, but there is no mention of the precedents here.

Volume II: Imperative Programming Languages

Volume II covers four languages: Fortran, C, Pascal, and Icon. Additionally, there is a short chapter on intermediate languages.

The Fortran 95 and Turbo Pascal chapters are average fare, and one of them contains the one unfortunate example of overenthusiastic advocacy in all four volumes: a section that begins, "Fortran 95 should be the language of choice for modern applications development." Apart from that, though, the authors do fair jobs of describing their languages. Glenn Grotzinger, who wrote about Turbo Pascal, is frank about Pascal's weaknesses (although it would have been interesting to see Kernighan's famous and excellent paper "Why Pascal Is Not My Favorite Programming Language" included as counterpoint).

Two chapters deal with C. The first is Dennis Ritchie's paper on the development of C. I had read the paper before but found it just as informative and entertaining the second time. Like Stroustrup, Ritchie is an excellent chronicler of his language's history, and his evaluation is even more honest and balanced. The second C chapter is a serviceable tutorial on C programming by Steve Summit, maintainer of the Internet C FAQ.

The final chapter of this volume, covering Icon, is one of the delights of the whole set. Written by Ralph Griswold, Icon's designer, it is very readable and manages to be thorough while concentrating the most attention on Icon's unusual features. The result is a chapter that succeeds in conveying what it is like to program in Icon – a unique, powerful, mind-expanding programming language.

Volume III: Little Languages and Tools

In terms of practicality, this volume is probably the jewel. The title is perhaps a little misleading; "Little Languages and Scripting Languages" may have been more appropriate. Many programmers will find the volume useful because it provides a nice introduction to a problem-solving strategy – domain-specific little languages – that is not well known in many communities. Others will find it useful as a source of information about the three most popular scripting languages.

A lot of languages are covered – too many to name here. The major ones are EQN, troff, awk, sed, SQL, Tcl/Tk, Perl, and Python. Mixed in are shorter treatments of several more of the common UNIX special-purpose languages and a bevy of little languages for music. While it's unfortunate that there's no explicit coverage of regular expressions (one of the archetypal little languages), they do surface in one form or another in at least seven of the twelve chapters in this book. (Come to think of it, that may be a more persuasive testimony to their utility than a chapter extolling their virtues.)

Chapter 1 is a reprint of Jon Bentley's wonderful *Programming Pearls* column of 1986. Titled "Little Languages," the chapter focuses on Pic, the picture-drawing language for troff. I recall that the first time I read this article, I thought that the choice of Pic was somewhat odd, but it soon became apparent that Pic is a perfect choice for illustrating the "little language" approach to problem solving. Itself a small language for a special purpose, Pic has been used as an intermediate language for several other languages that are even smaller or more specialized (such as grap, scatter, and chem). And in the other direction, the implementation of Pic makes use of three other little languages: lex, yacc, and make. Bentley does an extraordinary job. It's nice that this paper is reprinted here; the other book in which it has been printed, the second *Programming Pearls* volume, is apparently out of print.

Other chapters covering EQN and the troff typesetting system round out the discussion of a system for book preparation that seems to have been uniquely suited to the "little language" approach.

(Three of the first four chapters deal with typesetting and formatting software, and unlike the original versions, these chapters were not prepared with the software being described. The result, which amused me and no doubt frustrated the editor, is that the figures don't always convey what they should. Some of the figures are supposed to show the results of mistakes, but the people who set this book for printing occasionally decided to "fix" the mistakes. In one case, the "before" and "after" pictures are identical! Fortunately, it's usually clear from the text what should have been there. My advice is to smile and take this as evidence of the importance of the "little languages" approach that put so much control directly in the hands of authors.)

Paul Hudak contributed chapter 3, "Domain-Specific Languages." At first glance it seems to cover much the same ground as chapter 1, although there is slightly more emphasis on the process of choosing language primitives that match the domain. However, this chapter recommends a different approach to implementation. Whereas Bentley advocates building completely new little languages using specialized tools for language construction, Hudak recommends customizing an existing language with new procedures tailored for the problem domain. Functional languages work well for this purpose, and Hudak brings Haskell, the functional language he helped to design, into the discussion to demonstrate the approach.

Two of the chapters on scripting languages deserve mention, one for its failings and the other for its strengths. The "Perl Basics" chapter (one of two covering Perl) is unfortunately weak. Although I'm fond of Perl, it is definitely not an ordinary language, and it would be helpful for most readers to begin with some

discussion of Perl's philosophy and design principles, as an orientation. (This is especially true since the designer of Perl consciously and deliberately ignored many of the traditional principles of language design – uniformity and orthogonality, for example – that are recommended by authors of earlier chapters.) Instead, the chapter launches directly into a discussion of data types, and moves on from there into other nuts-and-bolts topics. There are plentiful examples, but few of them are long enough to provide much of a feel for what Perl programs are really like. In their eagerness to cover all of the basics, they give short shrift to some of Perl's distinguishing characteristics, such as the context-sensitive behavior of many operations. Some of the most distinctive and oft-used Perl features, including the `<FILEHANDLE>` input operator, are used without any explanation at all. I suspect few readers will understand from this chapter why so many people get excited about Perl.

The scripting language of the moment, it seems, is Python. Mark Lutz, a well-known Python advocate and the author of O'Reilly's *Programming Python* book, has done a terrific job of introducing Python. The coverage is broad enough that I felt as though I learned the whole language, although I know some details were omitted to keep the chapter short. The writing style is light and irreverent without being irritating. I had looked at Python on other occasions and had come away unimpressed, but Lutz almost made a believer out of me. I'm still in the camp of those who dislike Python's indentation-based syntax, but I'm coming around. (A cynic might make much of the fact that Lutz didn't mention the dependence on indentation until 30 pages into the chapter. Fortunately, I'm not a cynic.)

The final chapter of this volume returns to the topic of small, domain-specific languages. "Little Music Languages" was written by Peter S. Langston, a noted researcher on topics involving the combination of computers and music. The chapter introduces 16 little languages for describing and generating music. Most of them were designed for use by programs, rather than people (and a few even use binary encodings), but several can be read and understood easily by musicians, and most can be written or modified, if necessary, using a standard text editor. The chapter is fun to read and very informative. Furthermore, it is a significant contribution to the volume as a whole: To really understand the value of little languages it helps to see a broad sampling from vastly different domains. These languages are very different from the other languages in the book, and it is easy to see how the specialized languages are helpful to people working with computers and music. My only complaint is that there is no hint of how to acquire some of these tools to play with them, or whether that is even possible.

Volume IV: Functional and Logic Programming Languages

As a complete volume, this is the weakest of the four. It contains some excellent material, but ultimately it is a too narrow treatment of an important topic, because of the few languages that are represented. In fact, at some level only two languages are discussed: Lisp and Prolog. Several dialects of Lisp are mentioned, but seen as functional languages they are mostly the same. Prolog as the single example of logic languages is reasonable, but the spectrum of functional languages is much larger than what is shown here.

Emacs Lisp is used to introduce basic Lisp programming. I initially thought Emacs Lisp was a bizarre choice, but it actually makes sense. Emacs Lisp is almost certainly the most common Lisp implementation in the world, and readers who want to try Lisp programming for the first time will find it relatively easy to gain access to a copy of Emacs. Additionally, among currently popular Lisps, Emacs Lisp is especially close to the early MacLisp dialects that had so much influence on the Lisp community. The chapter on Emacs Lisp is an excellent introduction to the Lisp language. The author steers clear of Emacs Lisp's specialized types and facilities for text editing. Those things are interesting in their own right, but are wisely omitted from an introduction to Lisp as a general-purpose language.

Other chapters deal with Scheme, Guile (an embeddable Scheme interpreter), and CLOS – all dialects of Lisp. The chapters all have their virtues. The Scheme chapter is the best short description of Scheme I've read. The CLOS chapter is a terrific introduction to a system that is unusually powerful and flexible by any measure. And the discussion of Guile centers on the use of embeddable interpreters and command languages in larger applications. This is a powerful application-structuring mechanism, and although it is the niche that Tcl was designed for, the Tcl chapter in volume III never mentions it, so it is nice to see it discussed here.

For all their strengths, though, only the Scheme chapter tells us any more about functional programming than we learned in the Emacs Lisp chapter.

Prolog is the topic of the final chapter of the book. Of all the well-known languages in these four volumes, Prolog is the only one I've never written even a single line of code in, and my only prior reading on the topic consisted of a couple of sketchy articles in *Byte* magazine years ago. So I was interested in reading about what I had always thought of as a very limited, impractical language. The author shows how Prolog can do more than just evaluate assertions based on chains of reasoning. At the same time, he does not try to claim that Prolog is a good choice for every task. He concludes with several complete examples, including an implementation of Conway's "Game of Life" and a Lisp interpreter. Although I'm obviously no authority, this chapter seems to be an excellent introduction to logic programming.

Concluding Thoughts

In addition to the scale, any project like this comes accompanied by a couple of big pitfalls. First, any large, multi-author book is bound to suffer from uneven style and quality of writing. Second, no matter which languages are covered, some people will complain that their favorite has been omitted. (It might be more hazardous to prepare a *Handbook of UNIX Text Editors*, but not by much.)

For the most part, these books avoid the first problem. The various authors have different writing styles, of course, and cover their material at different levels of detail. With just a few exceptions, however, the quality of writing is high and the material is sound.

The second pitfall was not – in this reviewer's opinion – so well avoided. Although many interesting languages are represented here in addition to some old standbys, nearly every reader will find reason to wonder about some of the omissions. Despite my comments about the size of the project as it stands, my primary complaint is nevertheless that languages were omitted that should have been included. (There's precedent for this. Bjarne Stroustrup has noted that, during preparation of his book *The Design and Evolution of C++*, every review was of the form, "This book is too long . . . please add information on topics X, Y, and Z." So at least I'm in good company.)

I've mentioned the omission of languages more than once, and it may seem as though I'm belaboring a point. I do think these volumes are excellent as they stand and belong on the shelves of every computer-science library. However, when reading a review like this one, or scanning the table of contents in a bookstore, one tends to see what's present and not notice the omissions until later. So it seems reasonable to mention some languages that do not appear.

Volume I contains a good selection of object-oriented languages but would have been completed by discussion of a language that is not class-based, such as Self. Not only are such languages quite different and interesting when compared to more conventional object-oriented languages, recent research seems to indicate that prototype-based OO languages are in some sense more *fundamental*: Class-based OO languages can be modeled easily in terms of prototype-based languages, but doing it the other way around is more difficult.

Volume II contains some of the most surprising omissions: COBOL and PL/I are not included (although I confess that I personally didn't miss them). Additionally, I feel it would have been instructive to include Forth or some other stack-based language as an example of a very different approach, especially in conjunction with the chapter on intermediate languages.

Volume III is the most satisfying of the volumes in terms of completeness; it contains a broad cross-section of little languages and scripting languages. Still it's notable that the granddaddies of scripting languages (the Bourne shell and Rexx) are omitted. A discussion of macro processors such as m4 or the C preprocessor would have also been an interesting addition to the book.

Finally, Volume IV would have been greatly improved by broader coverage. APL warrants mention here (although I'm mindful that typesetting material about APL is expensive). There are actually several functional languages with some degree of current prominence; a chapter on Standard ML or Haskell would have provided some balance for the five Lisp chapters. And it would have been interesting to compare Prolog to Leda – a "multiparadigm" language that incorporates logic programming alongside imperative, functional, and object-oriented features.

To summarize: the *Handbook of Programming Languages* is an excellent compendium of information on a wide variety of languages. The overall quality is much higher than is typical for a multi-author work of this size. The only real problem is the omission of some important languages, but it is really a minor problem considering the huge scope of the books as they stand. The complete set will be of interest mainly to libraries and language buffs, but particular volumes will be very valuable to individuals.

USENIX news

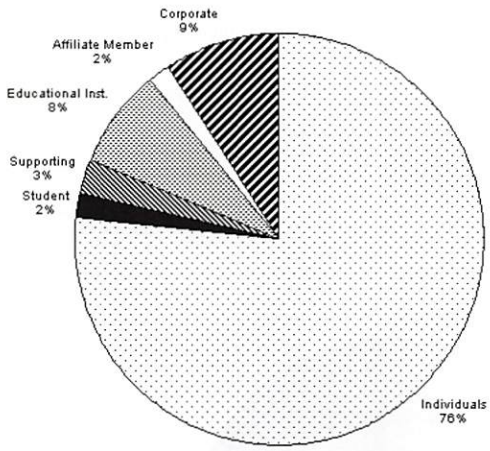
Member Dues

by Gale Berkowitz
USENIX Deputy Executive Director
<gale@usenix.org>

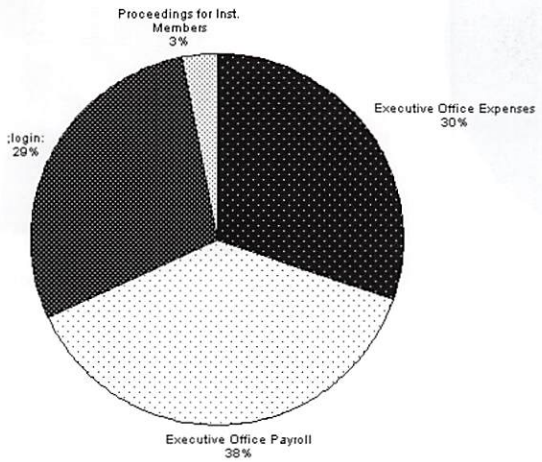
Here are a few charts that might help those of you who are curious about how your USENIX and SAGE dues are spent. The first shows the total dues income ($592,000 in 1998) divided by type of membership. The second chart then presents how those dues are spent. Note that income from our conferences covers all costs of the conference office personnel, exhibition, and marketing. The third chart shows how the executive office spends its money. The "other" category covers such items as taxes, licenses, bank charges, and miscellaneous expenses. The fourth chart indicates where most of the money allocated to good works and standards activities is spent ($812,000 in 1998). (See the USENIX Web site at <http://www.usenix.org/about/goodworks.html> for a description of our Good Works program.) These funds come from the income generated by the USENIX conferences and interest income from the Association's reserve fund.

Two charts deal with SAGE income ($212,000 in 1998) and direct expenses. Allocated expenses (staff and overhead) are not reflected in the direct expenses chart.

The 1998 Financial Statements for the USENIX Association follows these charts. These represent the Association's revenue and expenses for the year.



Membership Income Sources



Where Did Your '98 Membership Dues Go?

USENIX Member Benefits

As a member of the USENIX Association, you receive the following benefits:

Free subscription to ;login:,
the Association's magazine, published eight to ten times a year, featuring technical articles, system administration articles, tips and techniques, practical columns on Tcl, Perl, Java, and operating systems, book and software reviews, summaries of sessions at USENIX conferences, and reports on various standards activities.

Access to ;login: online
from October 1997 to last month.
<www.usenix.org/publications/login/login.html>

Access to papers
from the USENIX Conferences starting with 1993, via the USENIX Online Library on the World Wide Web.
<www.usenix.org/publications/library/index.html>

The right to vote
on matters affecting the Association, its bylaws, election of its directors and officers.

Optional membership
in SAGE, the System Administrators Guild.

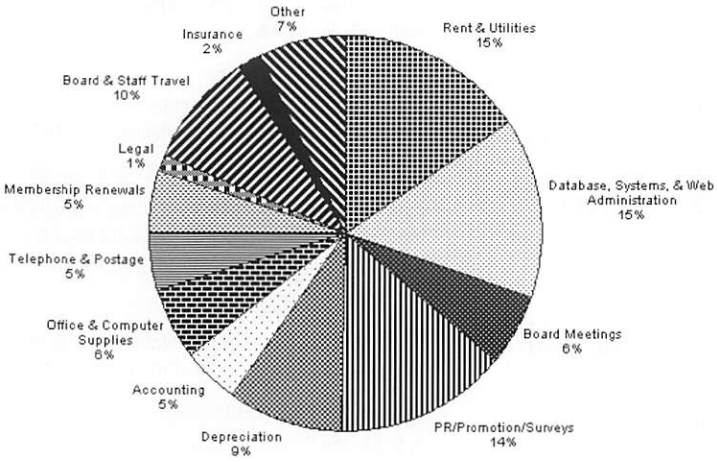
Discounts on registration fees
for all USENIX conferences.

Discounts
on the purchase of proceedings and CD-ROMs from USENIX conferences.

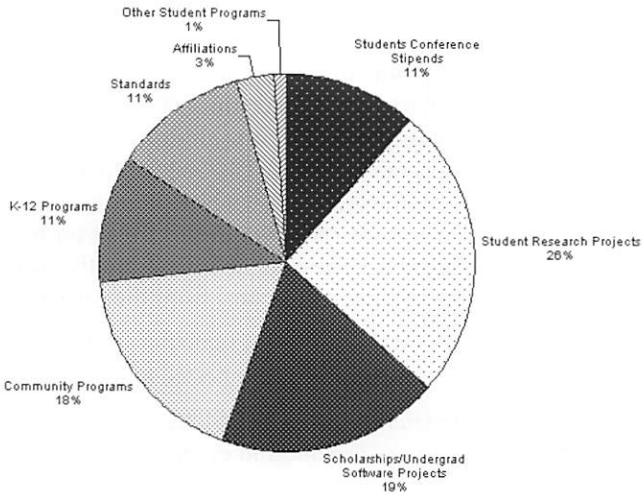
Savings
(see <http://usenix.org/membership/membership.html> for details)
10% off all Academic Press Professional books
10% off BSDI, Inc. "personal" products.
10% off Morgan Kaufmann Publishers books.
20% off New Riders/Cisco Press/MTP books.
10% off OnWord Press publications.
10% off The Open Group publications.
20% off O'Reilly & Associates publications.
$10.00 off Prime Time Freeware publications and software.
10% off Wiley Computer Publishing books.

Special subscription rates
(see <http://usenix.org/membership/membership.html> for details)
10% off subscription to any Cutter Information newsletter.
$45 subscription to *IEEE Concurrency* (regularly $88).
15% off subscription to *The Linux Journal.*
$5 off subscription to *The Perl Journal.*
20% off subscription to any Sage Science Press journals.
Free subscription to *Server/Workstation Expert.*
$10 off subscription to *SysAdmin Magazine.*

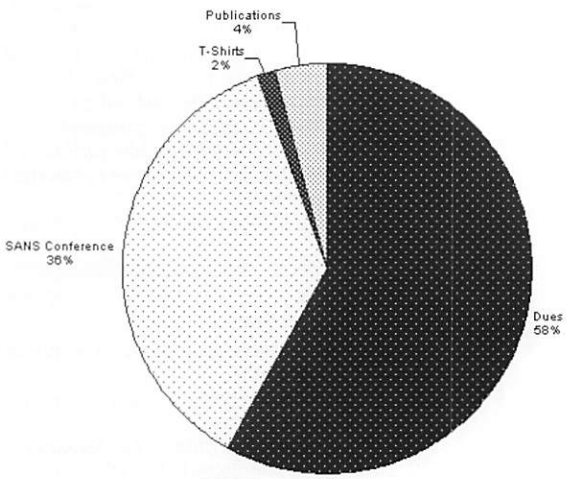
For more information regarding membership or benefits please contact
<office@usenix.org>
Phone: 510 528 8649



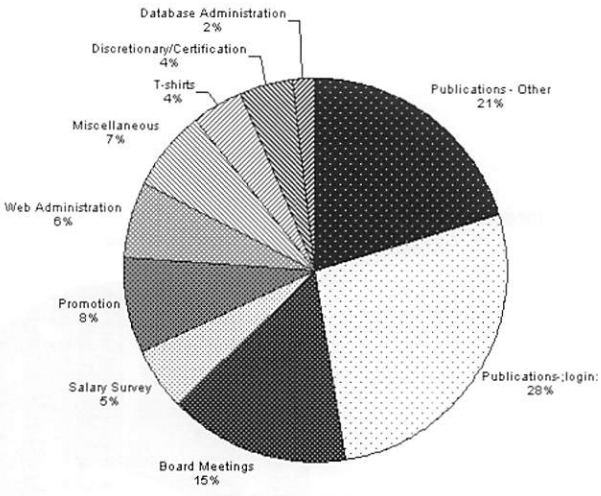
Executive Office Expenses



Good Works and Standards Spending



SAGE Income Sources



SAGE Direct Expenses

1998 Financial Statements

STATEMENTS OF FINANCIAL POSITION
As of December 31, 1998 and 1997

ASSETS	1998	1997
Current Assets		
Cash and cash equivalents	$ 2,059,597	$ 1,749,006
Receivables	60,714	48,897
Prepaid expenses	112,405	131,996
Inventory	22,311	30,324
Total current assets	2,255,027	1,960,223
Investment in securities [notes 3 & 4]	5,668,039	5,008,774
Property and Equipment		
Office furniture and equipment	487,824	448,179
Less: accumulated depreciation	(362,534)	(321,786)
Net Property and Equipment	125,290	126,393
Total assets	$ 8,048,356	$ 7,095,390

LIABILITIES AND NET ASSETS		
Liabilities		
Accrued expenses	$ 231,428	$ 123,511
Deferred revenue	0	200,613
Total liabilities	231,428	324,124
Net assets (unrestricted)	7,816,928	6,771,266
	$ 8,048,356	$ 7,095,390

STATEMENTS OF ACTIVITIES
For the Year Ended December 31, 1998 and the Thirteen Months Ended December 31, 1997

	1998	13 months 1997
REVENUES		
Conference revenue (Exhibits A & B)	$ 3,696,590	$ 3,018,821
Workshop revenue	729,267	630,750
Membership dues	533,129	580,238
SAGE membership dues & other income	190,952	201,016
Product sales	48,616	63,771
Interest on operating funds	68,730	76,426
Total revenues	5,267,284	4,571,022
OPERATING EXPENSES		
Conference expenses-direct (Exhibits A & B)	1,766,978	1,413,782
Workshop expenses-direct	500,677	453,061
Personnel & related benefits	1,016,021	920,782
Other general & administrative	528,079	489,854
Membership; login: /web	299,325	242,633
SAGE expenses	129,351	148,117
Product expenses	46,886	36,582
Projects & Good Works	812,167	750,112
Total operating expenses	5,099,484	4,454,923
Net operating surplus/(deficit)	167,800	116,099
NON-OPERATING ACTIVITY		
Donations	6,015	
Interest & dividend income-Reserve Fund	144,304	139,981
Gains & losses on marketable securities	788,667	141,774
Other non-operating income/(expense)	(17,327)	(29,590)
Investment fees	(43,797)	(51,009)
Net investment income & non-operating expense	877,862	201,156
Increase in net assets	1,045,662	317,255
Net assets, beginning of year	6,771,266	6,454,011
Net assets, end of year	$ 7,816,928	$ 6,771,266

STATEMENTS OF CASH FLOWS
For the Year Ended December 31, 1998 and the Thirteen Months Ended December 31, 1997

	1998	1997
CASH FLOWS FROM OPERATING ACTIVITIES		
Inrease in net assets	$ 1,045,662	$ 317,255
Adjustments to reconcile increase in net assets to net cash provided by/(used for) operating activities		
Depreciation	40,748	38,967
Decr/(Incr) in receivables	(11,817)	1,297
Decr/(Incr) in inventory	8,013	7,122
Decr/(Incr) in prepaid expenses	19,591	31,282
Incr/(Decr) in accrued expenses	107,917	(61,180)
Incr/(Decr) in deferred revenue	(200,613)	(161,887)
Total adjustments	(36,161)	(144,399)
Net cash provided by operating activities	1,009,501	772,206
CASH FLOWS PROVIDED BY/(USED FOR) INVESTING ACTIVITIES:		
Purchase of investments	(3,221,042)	(3,160,741)
Sale of investments	2,333,141	2,414,801
Distribution from reserve fund for good works	228,636	
Purchase of property & equipment	(39,645)	(68,794)
Net cash used for investing activities	(698,910)	(814,734)
Net change in cash & equivalents	310,591	(42,528)
Cash & equivalents, beginning of year	1,749,006	1,791,534
Cash & equivalents, end of year	$ 2,059,597	$ 1,749,006

20 Years Ago in USENIX

by Peter H. Salus
USENIX Historian
<peter@pedant.com>

The Toronto meeting had nearly 400 attendees. In the five years from May 1974 to June 1979, USENIX had acquired its name, and its meetings had waxed nearly twentyfold. UNIX had gone to 7th Edition.

Moreover, the ARPANET had grown to over 100 host sites, many of which used UNIX. Soon most of the hosts would be employing UNIX.

Bill Joy was shipping 2BSD (which contained software to be used with both V6 and V7), but would soon begin shipping 3BSD (Berkeley UNIX for the VAX, based on the 32V code, but with Berkeley utilities and a new virtual-memory system).

Remember: UNIX now ran on the Interdata and the VAX as well as the PDP-11s it was generally used for. The result was that folks worldwide saw it as a versatile, portable system.

At the beginning of the next year (1980), even the Department of Defense would realize that this was an advantage.

But prior to that, John Bass would organize another "western" meeting at SRI (it would be Brian Redman's first USENIX meeting) and, most importantly, Tom Truscott and Jim Ellis, students at Duke, came up with NETNEWS. Steve Bellovin, then at the University of North Carolina, wrote the first implementation.

USENET began as an exchange of information between UNC and Duke. But less than a year later, news leaked out.

USENIX Annual Awards

The USENIX Association, at its annual meeting in Monterey, presented the following awards:

The 1999 Lifetime Achievement Award

The X Window System Community at Large received USENIX's 1999 Lifetime Achievement Award, which recognizes

and celebrates singular contributions to the UNIX community in both intellectual achievement and unparalleled service. The citation says: "Presented in honor of a profound intellectual achievement and an unparalleled service to our Community." The following individuals are recognized as the Principal Recipients and Keepers of the Flame: Bob Scheifler, Jim Gettys, Phil Karlton, Ralph Swick, Keith Packard, and Smokey Wallace.

The 1999 Software Tools User Group Award

The 1999 award went to Udi Manber, for turning algorithms into tools for searching and resource discovery. Udi Manber is the Chief Scientist at Yahoo!. Before joining Yahoo! in 1998, he was a professor of Computer Science at the University of Arizona. He wrote more than 50 technical articles, 3 of which won best paper awards, co-developed Agrep, Glimpse, Harvest, and the Search Broker, and wrote a popular textbook on design of algorithms.

Further details can be found at <http://www. usenix.org/directory/awards.html> and at <http://www. usenix.org/directory/stug.html>.

Board Meeting Summary

by Ellie Young
Executive Director
<ellie@usenix.org>

Here is a sumary of some of the actions taken at the regular meeting of the USENIX Board of Directors held on June 26, 1999, in Monterey, California.

Policies

It was agreed that USENIX will restrict the rental of its mailing list and advertising in *;login:* to supporting members only. The only exception to the latter would be publishers offering a special discount offer to USENIX members.

A revised scheme for renumeration for tutorial speakers was approved.

Good Works

It was agreed to donate $2,000 to the Mark D. Weiser Excellence in Computing Scholarship Fund at UC Berkeley.

Conferences

Embedded Systems. It was agreed to schedule a second Workshop with some modifications. Geer will make a proposal for this, possibly including a wearable systems component.

Intrusion Detection. It was decided that there were other entities in the field with interest in running events on this topic. Rather than holding another separate event, it was suggested running a track at LISA.

Network Administration: USENIX will work with NANOG on possibly holding this event alongside one of their meetings.

Smard Card Technology: This workshop will be continued in 2001, possibly co-sponsored by CARDIS and co-located with a CardTech conference.

Security 2000: Young and Geer will solicit proposals to chair this event and also look into expanding the tutorial program.

LISA-NT and Win/NT. It was not certain whether these events would be repeated every 12 months. This will be discussed in July.

File Systems Workshop: It was suggested that it be co-located with OSDI 2000.

Atlanta Linux Showcase. When USENIX fully sponsors this conference in 2000,it may include a Linux Developers Workshop (a proposal from Tweedie, Yodaiken, and Ts'o will be forthcoming) and another Extreme Linux Workshop (a proposal from Beckman and Greenberg will be forthcoming.)

SANE 2000. It was agreed that USENIX would once again co-sponsor this event which is sponsored by NLUUG.

NordU 2000. It was agreed that USENIX would again co-sponsor this conference-with our affiliate, EurOpen.se in February in Malmö, Sweden. Parseghian will serve as liaison.

In Memoriam: Mark Weiser

In April, our community lost one of of its most beloved and admired figures. Mark Weiser, chief technologist at Xerox PARC, died after a short illness. In recognition of his many achievements a digital archive has been created at:
<http://www-sul.stanford.edu/weiser/>

As many of you know, Mark was best known for his work on ubiquitous computing. He was an excellent teacher, researcher, father, and friend to many of us.

Those interested in sending condolences to his family, or making contributions to the Mark D. Weiser Excellence in Computing Scholarship Fund at UC Berkeley, should visit the Xerox PARC memorial site at:
<http://www.parc.xerox.com/csl/members/weiser/>

WEB SITE

<http://www.usenix.org>

MEMBERSHIP

Telephone: 510 528 8649
Email: <office@usenix.org>

PUBLICATIONS

Jane-Ellen Long
Telephone: 510 528 8649
Email: <jel@usenix.org>

USENIX SUPPORTING MEMBERS

C/C++ Users Journal
Cirrus Technologies
Cisco Systems, Inc.
CyberSource Corporation
Deer Run Associates
Greenberg News Networks/ MedCast Networks
Hewlett-Packard India Software Operations
Internet Security Systems, Inc.
Microsoft Research
MKS, Inc.
Motorola Australia Software Centre
NeoSoft, Inc.
New Riders Press
Nimrod AS
O'Reilly & Associates
Performance Computing
Questra Consulting
Sendmail, Inc.
Server/Workstation Expert
TeamQuest Corporation
UUNET Technologies, Inc.
Windows NT Systems Magazine
WITSEC, Inc.

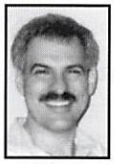
2nd Conference on Domain-Specific Languages

Sponsored by the USENIX Association
In cooperation with ACM SIGPLAN and ACM SIGSOFT
Sunday-Tuesday, October 3-5, 1999 - Omni Hotel, Austin, Texas, USA

Technical Program *Sunday, October 3, 1999*

Keynote Address

Towards More Natural Domain-Specific Languages
Brad A. Myers, *Human-Computer Interaction Institute, Carnegie Mellon University*



Most textual domain-specific languages were adapted from existing programming languages or were based on the intuition of the designer. The Natural Programming Project is developing general principles, methods, programming language designs, and environments that will provide a more scientific basis on which to base these designs, when the goal is a language that is easy to learn and effective for use by people who are not professional programmers. This talk will provide an overview of the Natural Programming approach and our results so far.

Brad A. Myers is a Senior Research Scientist in the Human-Computer Interaction Institute in the School of Computer Science at Carnegie Mellon University, where he is the principal investigator for various projects, including User Interface Software, Demonstrational Interfaces, Natural Programming, and the Pebbles PalmPilot Project. He is the author or editor of over 180 publications, including Creating User Interfaces by Demonstration *and* Languages for Developing User Interfaces, *and he is on the editorial board of five journals. His research interests include User Interface Development Systems, user interfaces, Programming by Example, programming languages for kids, Visual Programming, interaction techniques, window management, and programming environments. He belongs to SIGCHI, ACM, IEEE Computer Society, IEEE, and Computer Professionals for Social Responsibility.*

Testing and Experience Reports

Session Chair: James R. Larus, *Microsoft Research*

Using Production Grammars in Software Testing
Emin Gun Sirer and Brian N. Bershad, *University of Washington*

Jargons for Domain Engineering
Lloyd H. Nakatani, Mark A. Ardis, Robert G. Olsen, and Paul M. Pontrelli, *Bell Laboratories, Lucent Technologies*

Slicing Spreadsheets: An Integrated Methodology for Spreadsheet Testing and Debugging
James Reichwein, Gregg Rothermel, and Margaret Burnett, *Oregon State University*

Hot Research Review

Session Chair: Charles Consel, *Irisa/University of Rennes*

Domain-Specific Languages for Programming and Security in Active Networks
Carl A. Gunter, *University of Pennsylvania*

Active networks allow routing elements to be programmed by the packets passing through them., thereby enabling optimizations and extensions of current protocols as well as the development of fundamentally new protocols. To realize this flexibility, it is essential to provide models for programming and security that are easy to use while providing acceptable performance. In this lecture I will look at how domain-specific languages for programming active networks and describing security policies can be used to control global computation, avoid costly cryptographic operations, and enable formal specification and verification of essential properties.

Carl A. Gunter does research in the areas of programming languages and software engineering. He has contributed to foundations for the semantics of programming languages, type systems, and the design of programming languages. His research has also included contributions on computational logic, the representation of partial information, and mathematical models of software configuration dependencies. His current work focuses on active networks, security infrastructure systems, formal methods in software engineering, and liability analysis of software agreements and accidents.

Optimization and Extensibility

Session Chair: Mary Fernandez, *AT&T Labs—Research*

An Annotation Language for Optimizing Software Libraries
Samuel Z. Guyer and Calvin Lin, *University of Texas at Austin*

A Case for Source-Level Transformations in MATLAB
Vijay Menon and Keshav Pingali, *Cornell University*

Using Java Reflection to Automate Extension Language Parsing
Dale E. Parson, *Bell Laboratories, Lucent Technologies*

Technical Program *Monday, October 4, 1999*

Invited Talk

Language Technology for Performance and Security, or, Making Life Better, Not Just Easier
Peter Lee, *Carnegie Mellon University* and *Cedilla Systems Incorporated*

Modern languages strive to make programming easier. However, the real importance of modern language technology does not lie merely in ease of use. The same design principles that enable programs to be constructed more easily also lead to improved performance and safety. In order to illustrate this, I will give an introduction to proof-carrying code. PCC allows exceptionally high levels of performance and safety but for practical reasons depends critically on the design principles that underlie better languages for programmers.

Peter Lee is an Associate Professor of Computer Science at Carnegie Mellon University. His approach of applying theoretical ideas in programming language design to practical systems has led to numerous research contributions in the areas of programming language design, compiler technology, networking, and operating systems. Most recently, he has focused his attention on developing Proof-Carrying Code, a technique which uses program verification to enhance the performance and safety of mobile code. He is a principal investigator for the DARPA-sponsored Fox Project and is also the co-founder and president of Cedilla Systems Incorporated.

2nd Conference on Domain-Specific Languages

Sponsored by the USENIX Association
In cooperation with ACM SIGPLAN and ACM SIGSOFT
Sunday-Tuesday, October 3-5, 1999 - Omni Hotel, Austin, Texas, USA

DSLs and Monads

Session Chair: Paul Hudak, *Yale University*

DSL Implementation Using Staging and Monads
Tim Sheard, Zine-el-abidine Benaissa, and Emir Pasalic, *Oregon Graduate University*

Monadic Robotics
John Peterson and Greg Hager, *Yale University*

Hot Research Review

Session Chair: Todd Proebsting, *Microsoft Research*

A Methodology for Designing Domain-Specific Languages Using Program Specialization

Charles Consel, *Irisa/University of Rennes*

Domain-specific languages are mainly being developed in isolation. Furthermore, the difficult task of designing, structuring, and implementing a DSL requires expertise in multiple areas. DSLs can succeed only if development methodologies and tools are made available.

In this talk I give an overview of a methodology for developing DSLs. I also demonstrate how program specialization can map DSL interpreters into efficient (possibly just-in-time) compilers. The presentation is illustrated by concrete examples.

Charles Consel is a professor of computer science at the University of Rennes Irisa/Inria. He leads the Compose group at Inria. His group studies partial evaluation, a program transformation approach aimed at specializing programs with respect to given execution contexts. The work has been carried out with a program specializer for C called Tempo. This system has been successfully used in various applications such as operating systems and scientific code. A complementary research project is domain-specific languages: a software development approach that provides high productivity, easy maintenance, and improved safety (without giving up perfor-mance, thanks to partial evaluation). His work on programming languages, software engineering, and operating systems has led to many publications in major conferences and journals (POPL, PLDI, OOPSLA, ASE, SOSP, TOPLAS, ACM Surveys).

Embedded Languages

Session Chair: Michael Schwartzbach, *University of Aarhus*

Domain-Specific Embedded Compilers
Daan Leijen and Erik Meijer, *Utrecht University*

Verischemelog: Verilog Embedded in Scheme
James Jennings and Eric Beuscher, *Tulane University*

Technical Program *Tuesday, October 5, 1999*

Invited Talk

The Next 700 Markup Languages

Philip Wadler, *Bell Laboratories, Lucent Technologies*

XML (eXtensible Markup Language) is a magnet for hype: the successor to HTML for Web publishing, electronic data interchange, and e-commerce. In fact, XML is little more than a notation for trees and for tree grammars, a verbose variant of Lisp S-expressions coupled with a poor man's BNF (Backus-Naur form). Yet this simple basis has spawned scores of specialized sublanguages: for airlines, banks, and cell phones; for astronomy, biology, and chemistry; for the DOD and the IRS. Domain-specific languages indeed! There is much for the language designer to contribute here. Not least, as all this is based on a sort of S-expression, is there a role for a sort of Lisp.

Philip Wadler is a researcher at Bell Labs, Lucent Technologies. He is co-designer of the languages Haskell, Pizza, and GJ. He likes to spend his time on the border between theory and practice, looking for ways to use one to inform the other. He helped turn monads from a concept in algebraic topology into a way to structure programs in Haskell, and his work on GJ may help turn quantifiers in second-order logic into a feature of the Java programming language. He edits the Journal of Functional Programming *for Cambridge University Press and writes a column for* SIGPLAN Notices. *He was an ACM distinguished lecturer 1989–1993 and has been an invited speaker at conferences in Boulder, Brest, Gdansk, London, Montreal, New Haven, Portland, Santa Fe, Sydney, and Victoria.*

The Web, Data, and Collaboration

Session Chair: Jay Lepreau, *University of Utah*

Declarative Specification of Data-Intensive Web Sites
Mary Fernandez and Dan Suciu, *AT&T Labs—Research;* Igor Tatarinov, *North Dakota State University*

A Collaboration Specification Language
Du Li and Richard R. Muntz, *University of California, Los Angeles*

Hancock: A Language for Processing Very Large-Scale Data
Dan Bonachea, *University of California, Berkeley;* Kathleen Fisher and Anne Rogers, *AT&T Labs—Research*

Register now. On-line registration: *http://www.usenix.org/events/dsl99*

2nd USENIX Symposium on Internet Technologies & Systems

Sponsored by the USENIX Association
Co-sponsored by IEEE Computer Society Technical Committee on the Internet
Monday-Thursday, October 11-14, 1999 - Regal Harvest House Hotel, Boulder, CO, USA

Tutorial Program *Monday, October 11, 1999*

To meet your needs, the Tutorial Program at USITS '99 provides you with in-depth, immediately useful instruction in Internet techniques, effective tools, and best strategies.

Morning Tutorial Sessions

Web Application Security
Mark-Jason Dominus, *Consultant*

XML and Metadata for the Web
Neel Sundaresan, *IBM Almaden Research Center*

Afternoon Tutorial Sessions

Intrusion Detection and Network Forensics
Marcus J. Ranum, *Network Flight Recorder, Inc.*

An Introduction to Virtual Private Networks (Secure Networking)
Tina Bird, *Secure Networking Group*

Technical Program *Tuesday, October 12, 1999*

Keynote Address

E-Commerce—An Optimistic View
Udi Manber, *Yahoo! Inc.*

Will E-commerce change the world? Yes.

Do we know how? No.

Are there any interesting unsolved technical and other problems? In abundance.

Udi Manber, winner of the 1999 Annual Software Tools Users Group award, is Chief Scientist at Yahoo!. Before joining Yahoo! in 1998, he was a professor of computer science at the University of Arizona. He wrote more than 50 technical articles, 3 of which won best paper awards, co-developed Agrep, Glimpse, Harvest, and the Search Broker, and wrote a popular textbook on design of algorithms..

Shared Caching

Session Chair: P. Krishnan, *Bell Labs, Lucent Technologies*

Scalable Web Caching of Frequently Updated Objects Using Reliable Multicast
Dan Li and David R. Cheriton, *Stanford University*

Hierarchical Cache Consistency in a WAN
Jian Yin, Lorenzo Alvisi, Mike Dahlin, and Calvin Lin, *University of Texas at Austin*

Organization-Based Analysis of Web-Object Sharing and Caching
Alec Wolman, Geoff Voelker, Nitin Sharma, Neal Cardwell, Molly Brown, Tashana Landray, Denise Pinnel, Anna Karlin, and Henry Levy, *University of Washington*

Applications

Session Chair: Terence Kelly, *Microsoft Research* and *University of Michigan*

The Ninja Jukebox
Ian Goldberg, Steven D. Gribble, David Wagner, and Eric A. Brewer, *University of California at Berkeley*

Cha-Cha: A System for Organizing Intranet Search Results
Mike Chen, Jason Hong, James Lin, and Marti Hearst, *University of California at Berkeley*

A Document-based Framework for Internet Application Control
Todd D. Hodes and Randy H. Katz, *University of California at Berkeley*

Techniques

Session Chair: Eric A. Brewer, *University of California at Berkeley* and *Inktomi*

Sting: A TCP-based Network Measurement Tool
Stefan Savage, *University of Washington*

JPEG Compression Metric as a Quality-Aware Image Transcoding
Surendar Chandra and Carla Schlatter Ellis, *Duke University*

Visit our website: *http://www.usenix.org/events/usits99*

2nd USENIX Symposium on Internet Technologies & Systems

Sponsored by the USENIX Association
Co-sponsored by IEEE Computer Society Technical Committee on the Internet
Monday-Thursday, October 11-14, 1999 - Regal Harvest House Hotel, Boulder, CO, USA

Technical Program *Wednesday, October 13, 1999*

Proxy Implementation

Session Chair: Jeffrey Mogul, *Compaq Western Research Laboratory*

Secondary Storage Management for Web Proxies
Evangelos P. Markatos, Manolis G.H. Katevenis, Dionisis Pnevmatikatos, and Michail Flouris, *ICS—FORTH*

Compression Proxy Server: Design and Implementation
Chi-Hung Chi, Jing Deng, and Yan-Hong Lim, *National University of Singapore*

On the Performance of TCP Splicing for URL-Aware Redirection
Ariel Cohen, Sampath Rangarajan, and Hamilton Slye, *Bell Laboratories*

Prefetching

Session Chair: Geoffrey H. Kuenning, *Harvey Mudd College*

Prefetching Hyperlinks
Dan Duchamp, *AT&T Labs—Research*

Mining Longest Repeating Subsequences to Predict WWW Surfing
Jim Pitkow and Peter Pirolli, *Xerox PARC*

Architectures

Session Chair: David B. Johnson, *Carnegie Mellon University*

Active Names: Flexible Location and Transport of Wide-Area Resources
Amin Vahdat, *Duke University;* Michael Dahlin, *University of Texas at Austin;* Thomas Anderson and Amit Aggarwal, *University of Washington*

Person-level Routing in the Mobile People Architecture
Mema Roussopoulos, Petros Maniatis, Edward Swierk, Kevin Lai, Guido Appenzeller, and Mary Baker, *Stanford University*

A User's and Programmer's View of the New JavaScript Security Model
David M. Kristol and Alain Mayer, *Bell Labs*

Work-in-Progress Reports

Session Chair: Peter Honeyman, *CITI, University of Michigan*
Short, pithy, and fun, Work-in-Progress Reports introduce interesting new or ongoing work. If you have work you would like to share or a cool idea that's not quite ready for publication, contact the WIPS coordinator via email at *usits99wips@usenix.org*. We are particularly interested in presenting students' work. A list of topics is announced on-site.

Technical Program *Thursday, October 14, 1999*

Caching Policies

Session Chair: Katia Obraczka, *University of Southern California Information Sciences Institute*

Using Full Reference History for Efficient Document Replacement in Web Caches
Hyokyung Bahn, *Seoul National University;* Sam H. Noh, *Hong-Ik University;* Kern Koh and Sang Lyul Min, *Seoul National University*

Providing Dynamic and Customizable Caching Policies
J. Fritz Barnes and Raju Pandey, *University of California at Davis*

Exploiting Result Equivalence in Caching Dynamic Web Content
Ben Smith, Anurag Acharya, and Tao Yang, *University of California at Santa Barbara*

Server Implementation

Session Chair: Fred Douglis, *AT&T Labs—Research*

Efficient Support for Content-based Routing in Web Server Clusters
Chu-Sing Yang and Mon-Yen Luo, *National Sun Yat-Sen University*

Rapid Reverse DNS Lookups for Web Servers
William LeFebvre, *Group sys Consulting;* Ken Craig, *CNN Internet Technologies*

Connection Scheduling in Web Servers
Mark Crovella and Robert Frangioso, *Boston University;* Mor Harchol-Balter, *MIT Laboratory for Computer Science*

Register now. On-line registration: *http://www.usenix.org/events/usits99*

Announcement and Call for Papers **USENIX**

2000 USENIX Annual Technical Conference

http://www.usenix.org/events/usenix2000

June 18-23, 2000 **San Diego Marriott Hotel & Marina, San Diego, California**

Important Dates

Paper submission deadline: *November 29, 1999*
Notification to authors: *January 26, 2000*
Full papers due for editorial review: *March 28, 2000*
Camera-ready papers due: *April 25, 2000*

Conference Organizers

Program Committee:
Chair: Christopher Small, *Lucent Technologies—Bell Labs*
Ken Arnold, *Sun Microsystems*
Aaron Brown, *University of California at Berkeley*
Pei Cao, *University of Wisconsin*
Fred Douglis, *AT&T Labs—Research*
Edward W. Felten, *Princeton University*
Eran Gabber, *Lucent Technologies—Bell Labs*
Greg Minshall, *Siara Systems*
Yoonho Park, *IBM*
Vern Paxson, *International Computer Science Institute*
Liuba Shrira, *Brandeis University*
Keith A. Smith, *Harvard University*
Mark Zbikowski, *Microsoft*

Invited Talks Coordinators:
John Heidemann, *USC/Information Sciences Institute*
John T. Kohl, *Rational Software*

FREENIX Program Committee:
Chair: Marshall Kirk McKusick, *Author & Consultant*
Chris Demetriou, *The NetBSD Project*
David Greenman, *The FreeBSD Project*
Miguel de Icaza, *Instituto de Ciencias Nucleares, UNAM*
Ted Ts'o, *VA Linux Systems*
Stephen C. Tweedie, *Red Hat Software*

Overview

USENIX, founded twenty-five years ago, is the Advanced Computing Systems Association. Over the past quarter-century, the USENIX Association's membership has grown from its original core of UNIX users to a broad community of developers, researchers, and users with interests ranging from embedded systems to Tcl/Tk, from object-oriented programming to network administration, and from Internet technologies and electronic commerce to using, managing, and researching Windows NT. The USENIX 2000 Annual Technical Conference seeks to bring together this broad community under a single roof to share the results of their latest and best work, find points of common interest and perspective, and develop new ideas that cross and break boundaries.

The three-day technical session of the conference includes a track of refereed papers selected by the Program Committee; a track of Invited Talks by experts and leaders in the field; and FREENIX, a track of talks and paper presentation on freely available POSIX-based software and systems. Refereed papers are published in the Proceedings, which are provided to Technical Session attendees, along with materials from the Invited Talks and FREENIX presentations. Three days of tutorials precede the technical sessions with practical tutorials on timely topics.

Refereed Papers

The 2000 USENIX Technical Conference seeks to bring together the work, and the members, of the groups that make up the USENIX community. To that end, the Program Committee is interested in receiving submissions on a broad range of topics, including (but not limited to):

- Operating system and application structures for modern, commodity hardware, including extensible, embedded, distributed, and object-oriented systems.
- The impact of commodity hardware and software on the development of software systems.
- How the growing ubiquity of the Internet affects, and is affected by, the technological developments in the areas of electronic commerce, security, and heterogeneous and mobile computing.
- ActiveX, Java, CORBA, and other technologies that support mobile and reusable software components.
- The future of Tcl/Tk, Perl, and other scripting and domain-specific languages.
- Connecting, managing, and maintaining geographically distributed, heterogeneous networks of computers.

As at all USENIX conferences, papers that analyze problem areas, draw important conclusions from practical experience, and make freely available the techniques and tools developed in the course of the work are especially welcome.

Cash prizes will be awarded to the best paper and the best paper by a student.

How to Submit a Paper to the Refereed Track

Authors are required to submit full, complete papers by Monday, November 29, 1999. No papers will be accepted after 5:00 PM, Eastern time, Friday, December 3.

All submissions for USENIX 2000 will be electronic, in PostScript or PDF. Please follow the instructions for on-line submission at: *http://www.usenix.org/events/usenix2000/cfp/submit.html.*

Authors will be notified of receipt of submission via e-mail. If you do not receive notification, contact: *usenix2000-chair@usenix.org.*

Papers should be 8 to 12 single-spaced 8.5 x 11 inch pages (about 4000-6000 words), not counting figures and references. Papers longer than 14 pages and papers so short as to be considered extended abstracts will not be reviewed.

It is imperative that you follow the instructions for submitting a quality paper. Specific questions about submissions may be sent to the program chair via email to: *usenix2000-chair@usenix.org.*

A good paper will clearly demonstrate that the authors:

- are attacking a significant problem,
- are familiar with the literature,
- have devised an original or clever solution,
- if appropriate, have implemented the solution and characterized its performance using reasonable experimental techniques, and
- have drawn appropriate conclusions from their work.

Note: the USENIX Technical Conference, like most conferences and journals, requires that papers not be submitted simultaneously to more than one conference or publication, and that submitted papers not be previously or subsequently published elsewhere. Papers submitted to this conference that are under review elsewhere will not be reviewed. Papers accompanied by non-disclosure agreement forms can not be accepted, and will not be reviewed. All submissions are held in the highest confidentiality prior to publication in the Proceedings, both as a matter of policy and in accord with the U.S. Copyright Act of 1976.

Authors will be notified by January 26, 2000. All accepted papers will be shepherded by a program committee member through an editorial review process prior to final acceptance for publication in the proceedings.

FREENIX Track

FREENIX is a special track within the USENIX Annual Technical Conference. USENIX encourages the exchange of information and technologies between the commercial UNIX products and the free software world as well as among the various free operating system alternatives.

FREENIX is the showcase for the latest developments and interesting applications in freely redistributable software. The FREENIX forum includes Apache, FreeBSD, GNOME, GNU, Linux, NetBSD, OpenBSD, Samba, and more. The FREENIX track attempts to cover the full range of software which is freely redistributable in source code form and provides pointers to where the code can be found on the Internet. Marshall Kirk McKusick is serving as FREENIX program chair. FREENIX attendees also attend all of the USENIX Conference offerings and informal get-togethers.

Submissions to the refereed track are expected to represent mature work for which the authors are ready to fully describe the background, new ideas, experiments, and results of their work. By contrast, the FREENIX track seeks to gather reports on projects that are underway, but not yet finished. The purpose for the FREENIX papers is not as an archival reference, but rather a place to let others know about the project on which you are working and to provide a forum from which to expand your user base. We are looking for talks which advance the state of the art of freely redistributable software or otherwise provide useful information to those faced with deploying (and "selling") free software in the field.

Areas of interest include, but are not limited to:

- Operating system design
- Network design and implementation
- File system design
- Highly-available systems
- Highly-scalable systems
- Graphical user interface tools
- Desktop metaphors
- File and print systems
- System management tools
- Security
- Large scale system management
- Interesting deployments of free software
- How free software is being developed and managed today

Interesting applications of freely redistributable software might include: robotics and automation, clustering, wearable computers, embedded systems, high-speed networking, studio graphics, and audio processing.

How to Submit to the FREENIX Track

Authors are required to submit one to three page summaries by Monday, November 29, 1999. No submissions will be accepted after 5:00 PM, Eastern time, Friday, December 3.

All submissions for USENIX FREENIX 2000 Track will be electronic. Please use this web form to send your submission: *http://www.usenix.org/events/usenix2000/cfp/freenix-submit.html.*

Authors will be notified of receipt of submission via e-mail. If you do not receive notification, contact: *freenix2000-chair@usenix.org.*

You may submit a full paper, however we expect that most submissions will be 1–3 page summaries of your work to date. Please provide enough detail to let us know what you are doing. In no event should you submit a description in excess of 14 pages.

Specific questions about submissions may be sent to the program chair via email to: *freenix2000-chair@usenix.org.*

A good submission will clearly demonstrate that the authors:

- are attacking a significant problem,
- are actively working on a solution, and
- have made enough progress to have useful information to report.

In particular, the FREENIX track is not for people that are thinking about doing some project, but have not yet started it. Such talks are better presented in the Work-In-Progress (WIP) session.

Authors will be notified by January 26, 2000. All accepted submissions will be expected to produce a written report for the proceedings by the March 28, 2000 deadline. These reports need not be polished papers as would be submitted to the refereed track, but they should describe work that has been completed as of the time of their submission. The purpose of your paper is to let readers and attendees know what you are doing. Your talk at the conference may describe not only what is in your paper but also the work completed between the time that the paper is submitted and the conference is held. Members of the program committee will help shepherd authors through the writing process prior to final acceptance for publication in the proceedings.

Submitting a Tutorial Program Proposal

On Sunday-Tuesday, June 18-20, USENIX's well-respected tutorial program offers intensive, immediately practical tutorials on topics essential to the use, development, and administration of advanced computing systems. Skilled instructors, who are hands-on experts in their topic areas, present both introductory and advanced tutorials covering topics such as:

- High availability and quality of service
- Distributed, replicated, and web based systems
- System administration and security
- Embedded systems
- File systems and storage systems
- Interoperability of heterogeneous systems
- Operating systems (Linux, BSD*, NT, etc.)
- Application development (threads, Perl, etc.)
- Intrusion detection and prevention
- Internet security
- Mobile code and mobile computing
- New algorithms and applications
- Systems application configuration and maintenance
- Personal digital assistants
- Security and privacy
- Web-based technologies

To provide the best possible tutorial slate, USENIX continually solicits proposals for new tutorials. If you are interested in presenting a tutorial, contact: Dan Klein, Tutorial Coordinator, Phone: 1.412.422.0285, Email: *dvk@usenix.org*

Submitting an Invited Talk Proposal

These survey-style talks given by experts range over many interesting and timely topics. The Invited Talks track also may include panel presentations and selections from the best presentations at recent USENIX conferences.

The Invited Talks coordinators welcome suggestions for topics and request proposals for particular talks. In your proposal state the main focus, including a brief outline, and be sure to emphasize why your topic is of general interest to our community. Please submit via email to *usenix2000-it@usenix.org.*

Work-in-Progress Reports

Do you have interesting work you would like to share, or a cool idea that is not yet ready to be published? The USENIX audience provides valuable discussion and feedback. We are particularly interested in presentation of student work. To request a WIP slot, send email to *usenix2000-wips@usenix.org.*

Birds-of-a-Feather Sessions (BOFs)

The always popular evening BOFs are very informal, attendee-organized gatherings of persons interested in a particular topic. BOFs may be scheduled at the conference or in advance by contacting the USENIX Conference Office at 1.949.588.8649 or via email to *conference@usenix.org.*

USENIX Exhibition

In the exhibition, the emphasis is on serious questions and feedback. Vendors will demonstrate the features and technical innovations which distinguish their products. For more information, please contact: Dana Geffner, USENIX Exhibition Coordinator, Phone: 1.831.457.8649, Email: *dana@usenix.org*

motd

by Rob Kolstad

Dr. Rob Kolstad works as program manager organizing computer security conferences. Longtime editor of *;login:*, he is also head coach of the USENIX-sponsored USA Computing Olympiad.

<kolstad@usenix.org>

Finishing Things

I had an interesting typesetting experience lately. On Valentine's Day, February 14, I was handed the "final final copy'" for a book that needed to be published by July 4. "This text is perfect," said the co-author. "It is ready to be typeset and published."

Of course, this turned out not to be the strict and total truth. A few changes here, a few modifications there, an adjustment or two, followed by the revelation: the book's editor intended to use the galley proofs for extensive editorial change! The book was finally ready for serious typesetting by the time May arrived.

I asked my friend how this could happen. He pointed out that his co-authors assured him that their parts were "finished." He shared his belief with me, and thus the spiral of change after change began.

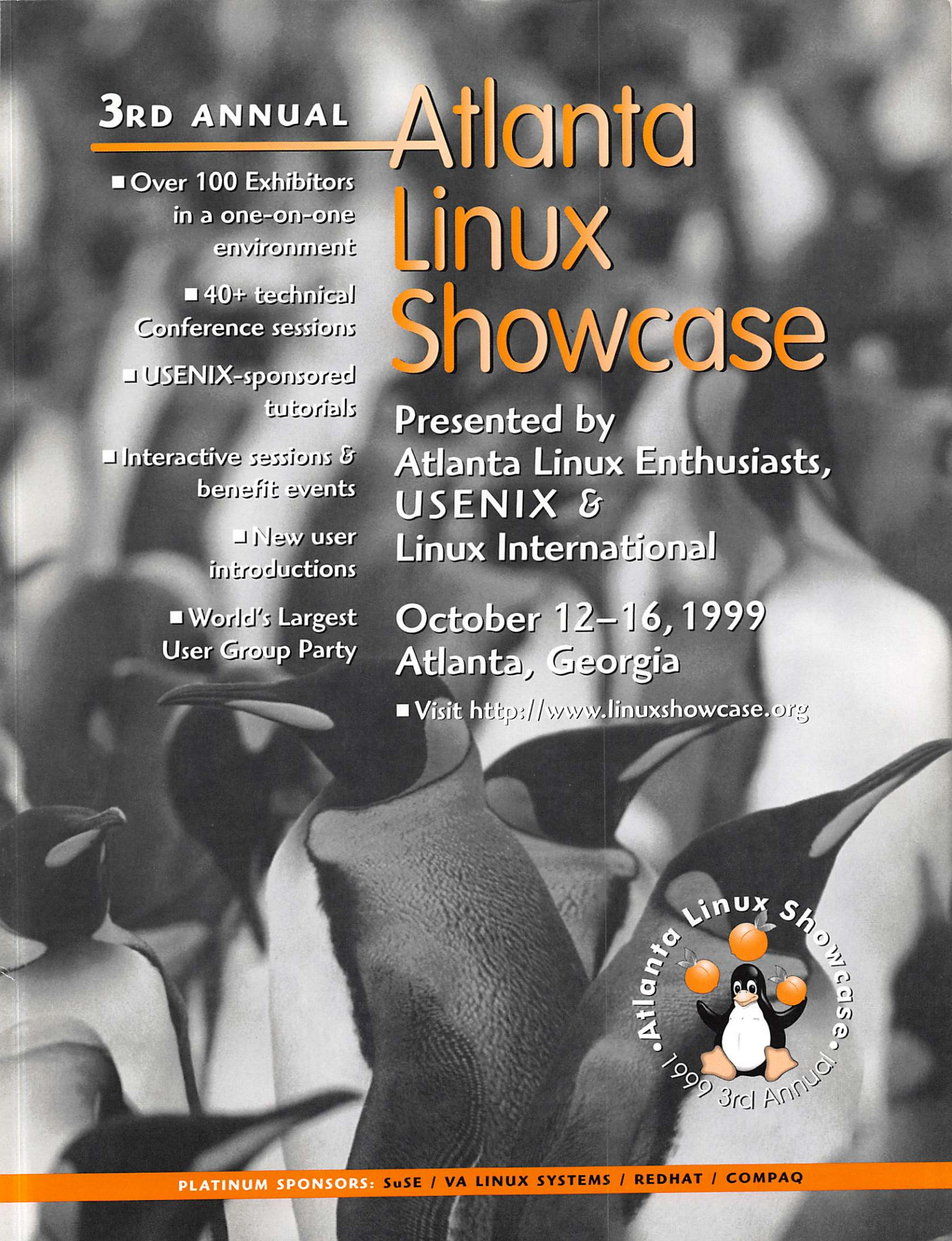
I think that finishing projects is hard! Throughout work and home life, I see projects that are begun – or even 90% finished – that never quite make it to the end of the creativity/implementation cycle. Maybe it is part of human nature to avoid the tremendous work at the end of a project's cycle?

I am fairly sure that, for me, deadlines hasten completion of projects. This editorial, though a few days late, has a very strict deadline. When the magazine heads out to the printer, the deadline has truly passed and few things can resurrect it. My new boss is great with deadlines – he sets them all the time. They surely do provide clear focus and an ability to see to the "end of the tunnel."

I've had a good run, I think, at completing work-related projects. In my own life, though, I have many not-quite-complete projects. I am building an electronic interface for a video switcher – it's over halfway finished. Another interface for an audio switcher is 80% finished and has been for almost a year! I feel terrible about these things, even as I contemplate starting other, more exciting projects.

I think finishing things is the epitome and culmination of the creative process. For me, it is surely among the most rewarding of activities. Yet I avoid it sometimes for its finality and the level of pain involved in going from 95% to 100% complete. I wish I knew why.

Best of luck to you for your projects this year; may their completions bring you joy and fulfillment.


3RD ANNUAL
Atlanta Linux Showcase
■ Over 100 Exhibitors in a one-on-one environment
■ 40+ technical Conference sessions
■ USENIX-sponsored tutorials
■ Interactive sessions & benefit events
■ New user introductions
■ World's Largest User Group Party
Presented by Atlanta Linux Enthusiasts, USENIX & Linux International
October 12–16, 1999
Atlanta, Georgia
■ Visit http://www.linuxshowcase.org
Atlanta Linux Showcase
1999 3rd Annual
PLATINUM SPONSORS: SuSE / VA LINUX SYSTEMS / REDHAT / COMPAQ

CONNECT WITH USENIX

MEMBERSHIP AND PUBLICATIONS

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: <office@usenix.org>

WEB SITE

http://www.usenix.org

EMAIL

login@usenix.org

COMMENTS? SUGGESTIONS?

send email to jel@usenix.org

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to <login@usenix.org> or through the postal system to the Association office.

Send SAGE material to <tmd@usenix.org>. The Association reserves the right to edit submitted material. Any reproduction of this magazine in its entirety or in part requires the permission of the Association and the author(s).

The closing dates for submissions to the next two issues of *;login:* are October 5, 1999, and December 1, 1999.

USENIX The Advanced Computing Systems Association

;login:

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send Address Changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710